Monitorització automàtica de bases de dades *NoSQL* per entorns *batch*

Grau en Enginyeria Informàtica – Enginyeria de Computadors

27/04/22

Autora: Anna Riera Ferrón

Director: Juan José Costa Prats (DAC)

Codirectora: Yolanda Becerra Fontal (DAC)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

FIB

RESUM

En informàtica el temps és or i cada segon compta, és per això que es vol que qualsevol execució sigui el més eficient possible.

En aquesta idea és on es recolza aquest projecte, que té l'objectiu d'oferir una aplicació agradable per a l'usuari i de codi obert que permeti monitoritzar bases de dades *Cassandra* i visualitzar-ne els paràmetres mitjançant gràfiques i que, gràcies a les dades extretes amb aquesta, se'n pugui millorar el rendiment de la base de dades.

RESUMEN

En informática el tiempo es oro y cada segundo cuenta, es por eso que se quiere que cualquier ejecución sea lo más eficiente posible.

En esta idea se apoya este proyecto, que tiene como objetivo ofrecer una aplicación agradable para el usuario y de código abierto que permita monitorizar bases de datos *Cassandra* y visualizar sus parámetros mediante gráficas y que, gracias a los datos extraídos de esta, se pueda mejorar el rendimiento de la base de datos.

ABSTRACT

In computing, time is money and every second counts, that is why we want any execution to be as efficient as possible.

This project is based on this idea, which aims to offer an open source and pleasant application for the user that allows monitoring *Cassandra* databases and visualizing their parameters through graphs and that, thanks to the data extracted from it, can be improved the database performance.

Índex

Índex d’imatges

Índex de taules

1. Introducció

1.1. Context

Com diu el refrany, el temps és or. En termes d'informàtica, cada segon compta i aconseguir reduir el temps d'execució d'un programa, ni que sigui uns segons, ja val la pena.

Per poder millorar els temps d'execució cal poder extreure estadístiques i saber què passa durant l'execució d'aquests programes, per detectar els punts dèbils i reforçar-los. Aquí és on entra aquest treball.

Per aquest projecte, s'utilitzarà, en remot, un *clúster* anomenat *Minerva* que utilitza la base de dades *Apache Cassandra*. Aquesta és de codi obert, distribuïda i *NoSQL*, i es caracteritza per funcionar mitjançant diferents nodes independents, que interconnectats entre si poden funcionar en conjunt. (1)

Aquesta característica la dota de molta flexibilitat, eficiència i possibilitat de creixement, però també fa que calgui recopilar dades de rendiment per assegurar que tots els nodes funcionen a part iguals i de la manera més eficient.

En aquesta recopilació de rendiment és on se centra aquest treball sobre la monitorització automàtica de bases de dades *NoSQL* per entorns *batch*.

1.2. Definicions de conceptes

En aquest apartat es defineixen conceptes propis del tema que tracta el treball, que són importants per entendre el desenvolupament d'aquest i que poden ser desconeguts pel lector.

- **Base de dades:** És un conjunt d'informació o dades, ben estructurades i organitzades per tal que siguin accessibles i ràpides de consultar que habitualment s'emmagatzemen digitalment. (2)

- **Base de dades distribuïda:** A priori semblen bases de dades comunes i per a l'usuari base es comporten com a tal, però en realitat són un conjunt de bases de dades, emmagatzemades en diferents ordinadors que treballen juntes i poden ser accedides i modificades de manera simultània. (3)

- **Base de dades *NoSQL*:** Les bases de dades *NoSQL* són aquelles que no utilitzen llenguatge *SQL* per realitzar les consultes. El terme *NoSQL* sorgeix de l'anglès *Not Only SQL.*

  Tot i que totes les bases de dades *NoSQL* comparteixen aquesta característica, no són totes iguals, n'existeixen set tipus diferenciats.

  Cassandra és del tipus clau-valor les quals, com el seu nom indica, utilitzen una clau com a identificador únic per a referenciar un conjunt de valors. (4)

- **Entorn *batch*:** Un entorn *batch* és aquell que utilitza processament *batch* per funcionar. El processament *batch* és aquell que executa programes que no requereixen la interacció d'un usuari durant la seva execució. (5)

- ***Clúster:*** És un conjunt de computadores que, connectades mitjançant una xarxa, treballen en conjunt com si fossin només una, repartint-se la feina de processament per a ser més eficients.

1.3. Descripció del problema

Actualment, *Cassandra* disposa ja d'una eina de monitorització inclosa anomenada *Nodetool* (6). Aquesta permet, mitjançant la línia d'ordres del terminal, extreure dades de rendiment i funcionament de la base de dades durant l'execució d'un programa.

El problema d'aquesta és, que és molt tosca, funciona només mitjançant el terminal, la qual cosa per usuaris menys experts pot suposar un problema, les dades de l'execució s'han d'extreure a mà durant l'execució i a més la informació que t'aporta és numèrica i una mica críptica si no saps ven bé el que busques.

Per altra banda, també es pot monitoritzar *Cassandra* amb l'eina *JConsole*, que tot i que a diferència de *Nodetool*, si disposa d'interfície gràfica, aquesta no és gaire amigable, ja que t'aclapara amb un munt d'informació que potser l'usuari no necessita, i a més, a causa d'aquesta gran quantitat d'informació, és molt difícil trobar alguna cosa en concret.

És per això que cal l'eina que es pretén desenvolupar en aquest projecte. L'objectiu d'aquesta nova eina és monitoritzar de forma automàtica, partint de l'eina *JConsole*, les execucions dels programes que corrin sobre *Cassandra*, sense necessitat d'escriure a mà línies d'ordres durant l'execució dels programes i podent consultar els resultats un cop acabada l'execució, d'una forma senzilla.

A més es vol desenvolupar una interfície gràfica més "amigable" amb botons i pestanyes, i únicament amb els atributs que realment es vulguin monitoritzar, que faciliti les consultes i que mostri gràfiques que permetin extreure conclusions més clares de les dades obtingudes.

1.4. *Stakeholders*

Els *Stakeholders* que es troben llistats a continuació són tots aquells individus que es beneficien o els pot interessar aquest projecte.

- **Usuaris que utilitzin *Cassandra*:** Tots aquells usuaris tant de Minerva com d'altres llocs, que necessitin monitoritzar *Cassandra* disposaran d'una nova eina per a fer-ho i, per tant, es beneficien de forma directa d'aquest projecte.

- **Directors del TFG:** Els directors, Juan José Costa Prats i Yolanda Becerra Fontal, ambdós usuaris que utilitzen Minerva i que a més seran els responsables de supervisar aquest projecte.

- **Autora del TFG:** Jo, Anna Riera Ferrón, autora d'aquest TFG, soc un dels principals agents implicats, ja que soc qui desenvoluparà i testejarà l'eina de monitorització i qui en documentarà el seu desenvolupament.

2. Justificació

Com ja s'ha comentat en altres apartats, *Cassandra* ja disposa de dues eines integrades de monitorització anomenades *Nodetool* i *JConsole* que tot i ser funcionals i permetre recollir dades d'execució, són tosques, difícils d'aprendre a utilitzar i a més, en el cas de *Nodetool*, limitada, ja que no permet una visualització gràfica de les dades ni una consulta posterior a l'execució.

Per altra banda, és cert que existeixen altres eines de monitorització per a *Cassandra*, com per exemple *Datadog*, que permet moltes de les funcionalitats que en aquest projecte es pretenen implementar, com la visualització de gràfiques o la interfície agradable, però que no és gratuïta i que no dona accés al codi font. (7)

Tot i existir altres eines com les esmentades, la que es vol desenvolupar al llarg d'aquest projecte continua tenint un sentit. En primer lloc, es vol una eina gratuïta i es vol poder accedir al codi font, per poder fer adaptacions o millores.

A més, *Cassandra* consta de milers de paràmetres de configuració que influeixen en el rendiment dels accessos i tot i que és relativament senzill posar en marxa un *clúster* de *Cassandra*, fer que sigui eficient pot resultar complicat. És per això que és necessària una eina que orienti a l'usuari a l'hora d'establir les configuracions mitjançant unes mètriques que es puguin relacionar amb els esmentats paràmetres.

També es vol crear una aplicació que sigui més fàcil d'usar i més visualment atractiva que les integrades amb *Cassandra*.

Finalment i més important, aquest projecte té un objectiu docent, es vol que al llarg del desenvolupament s'aprenguin noves habilitats i coneixements que no es tenien a l'inici. Tant de monitorització de bases de dades, com de programació Java, com de programació d'interfícies.

És per tots els motius esmentats que té sentit dur a terme aquest projecte.

3. Abast

3.1. Objectius i subobjectius

L'objectiu principal d'aquest projecte, com ja s'ha comentat en apartats anteriors, és crear una aplicació agradable per l'usuari que monitoritzi les execucions de programes a *Cassandra* i n'extregui dades que es mostrin de manera visual a l'usuari per tal que aquest en pugui treure conclusions i pugui millorar-ne el rendiment.

Per a poder dur això a terme ens calen els següents subobjectius:

- Permetre que l'usuari pugui recopilar dades d'execució de forma senzilla mitjançant una interfície gràfica.

- Permetre que l'usuari consulti aquestes dades un cop acabada l'execució.

- Permetre que l'usuari visualitzi les dades obtingudes mitjançant gràfiques i en pugui extreure conclusions.

3.2. Requisits

Els requisits no funcionals que ha de complir l'aplicació perquè s'adapti adequadament al que es busca amb el projecte són:

- **Aparença:** L'eina ha de ser atractiva per tal que l'usuari se senti còmode utilitzant-la.

- **Usabilitat:** L'eina ha de ser fàcil d'utilitzar per als usuaris i ha de ser senzill trobar les funcionalitats de les quals disposa.

3.3. Riscos i Obstacles

Al llarg de qualsevol projecte poden sorgir varietat de problemes i imprevistos que modifiquin la planificació i objectius finals d'aquest. A continuació es llisten els possibles riscos i obstacles que es poden donar al llarg del desenvolupament d'aquest TFG:

- **Inexperiència en les tecnologies i recursos utilitzats:** La majoria d'eines i entorns utilitzats per desenvolupar aquest treball no els he utilitzat mai, per tant, es pot requerir un temps addicional d'aprenentatge i investigació que pot provocar retards o inconvenients en el desenvolupament i la planificació.

- ***Bugs*:** En un projecte que requereix programació sempre es corre el risc de trobar *bugs* que endarrereixin i dificultin el desenvolupament.

- **Problemes de disponibilitat del *hardware*:** En estar desenvolupant l'eina mitjançant *hardware* remot, el projecte depèn per complet del bon funcionament d'aquest. És per això que, tot i ser molt poc probable, si el *hardware* fallés, o tingués algun problema que l'inutilitzes durant un període de temps, aquest projecte patiria retards o fins i tot, en cas de ser un període molt extens, podria arribar a no poder-se realitzar.

4. Metodologia

4.1. Metodologia de treball

Escollir una bona metodologia de treball és important per a portar un bon ritme i un bon control sobre el desenvolupament i poder reaccionar a temps en cas de trobar-se amb inconvenients.

Per aquest projecte s'ha utilitzat la metodologia *Agile*, la qual permet portar un control continu mitjançant petites tasques i reunions setmanals amb els directors del TFG i fa que el desenvolupament sigui molt més flexible i es pugui adaptar fàcilment a qualsevol imprevist.

4.2. Eines de desenvolupament

Per a dur a terme el projecte s'ha fet ús de les següents eines:

- ***JConsole* i *MBeans*:** L'eina integrada *JConsole* s'usa per extreure, mitjançant els *MBeans* de *Java*, les dades sobre les execucions a *Cassandra*.

- ***Cassandra*:** Cal una base de dades *Cassandra* per a poder fer el *testing* de la monitorització que durà a terme l'eina desenvolupada durant el projecte. Concretament, s'usaran els nodes de Minerva.

- ***IntelliJ IDEA*:** S'usa *IntelliJ IDEA* per a desenvolupar el projecte, tant l'extracció de dades com la interfície gràfica amb la qual accedir i consultar les dades extretes.

- ***JfreeChart*:** Per a crear les gràfiques que mostrin les dades i resultats es fa ús de la llibreria de codi obert de *Java JfreeChart*.

- ***GitHub*:** És el sistema de control de versions que s'està fent servir durant el desenvolupament del projecte per a mantenir organitzat el codi en diferents versions recuperables per si sorgeix algun problema.

- **Mètode *Agile*:** Mètode utilitzat per organitzar el desenvolupament del treball, mitjançant reunions setmanals i una planificació inicial dividida en *sprints* que permeten el control periòdic adequat per un bon desenvolupament.

5. Descripció de les tasques

La durada d'aquest projecte ha anat (de forma excepcional, ja que s'ha necessitat una extensió) del 20 de setembre de 2021 amb l'inici de la gestió del projecte, fins a la setmana del 25 al 30 d'abril de 2022 on es farà la lectura. Això fa un total d'aproximadament set mesos de duració tot i que com s'explicarà més endavant, això no és del tot cert, ja que només uns quatre mesos han estat realment invertits en el resultat final.

Per tal de poder entendre bé la planificació següent cal saber que tot i haver-hi una planificació inicial, degut a inconvenients personals i de desenvolupament, s'han requerit més hores de les esperades i és per això que s'ha allargat l'entrega d'aquest treball uns mesos.

Els problemes de desenvolupament són que es va haver de canviar d'eina per a recollir les dades de *Cassandra*, concretament de *Nodetool* a *JConsole* i els *MBeans* per a poder accedir-hi, ja que *Nodetool* era massa lent en els seus accessos, és per això que tota la feina feta per a *Nodetool* no s'ha pogut aprofitar i per simplicitat considerarem tot aquest període dedicat a *Nodetool* com una sola tasca N que va durar 30 h i considerarem que el desenvolupament de l'aplicació va tornar a començar de zero el 7 de gener, tornant de les vacances de Nadal.

En primer lloc, tenim, del 20 de setembre al 18 d'octubre, la gestió del projecte que va durar un més i va tenir una càrrega de treball de 75 h.

A continuació hi ha l'etapa de desenvolupament, *testing* i documentació, aquesta s'ha dut a terme entre el 7 de gener de 2022 i el 20 d'abril de 2022. Aquesta ha tingut una durada de 303 h tenint en compte que en mitjana s'han fet 3 hores al dia de feina.

Finalment ens quedarà aproximadament una setmana per preparar la presentació del projecte, a la qual se li dedicaran 21 h.

A més s'han fet un total de 27 reunions d'una hora amb els directors del projecte, el que fa un total de 27 h addicionals, i un estudi i anàlisi de l'estat de l'art inicial que va durar 10 h.

Així doncs, en total el projecte ha tingut una duració estimada de 466 hores.

A continuació es descriuran detalladament totes les tasques que s'han dut a terme al llarg del projecte.

En totes les tasques descrites s'utilitzen com a recursos materials el meu ordinador personal amb connexió a internet, l'accés remot a *Minerva* mitjançant *Putty*, *IntelliJ IDEA*, *JfreeChart* i *GitHub*.

Com a recursos humans estic només jo, l'autora del projecte, i els directors d'aquest.

També cal destacar, que tot i que els *sprints* 2 i 3 estan separats i planificats un darrere l'altre, al final s'han treballat bastant en paral·lel, ja que per poder mostrar les gràfiques calia incrustar-les dintre la interfície gràfica, així i tot, i per claredat, com que el càlcul final d'hores és el mateix, s'ha mantingut aquesta planificació.

Finalment, cal comentar que les hores de la majoria de tasques establertes inicialment s'han reorganitzat per donar més pes a allò que ho ha anat requerint. Concretament, s'ha donat moles més hores a la implementació dels *sprints* i l'aprenentatge de les eines requerides i se li ha restat pes a la resta de tasques.

5.1. Tasques de Gestió

- **G1 – Abast i contextualització del projecte**
    - Descripció: Definir l'abast, la contextualització i la justificació del projecte.
    - Durada: 25 h (20/9/21 - 28/9/21)
    - Dependències: -

- **G2 – Planificació temporal**
    - Descripció: Definir les tasques, fer un diagrama de Gantt i planificar la gestió de riscos.
    - Durada: 8 h (29/9/21 - 4/10/21)
    - Dependències: G1

- **G3 – Gestió econòmica i sostenibilitat**
    - Descripció: Fer un pressupost i un informe de sostenibilitat.
    - Durada: 9 h (5/9/21 - 11/10/21)
    - Dependències: G2

- **G4 – Document final**
    - Descripció: Unió dels tres apartats anteriors i correcció d'errors que poguessin tenir.
    - Durada: 18 h (12/10/21 - 18/10/21)
    - Dependències: G3

- **E – Estudi i anàlisi de l'estat de l'art**
    - Descripció: Investigació dels programes que ja existeixen com *Nodetool*, *Datadog* o *JConsole*.
    - Durada: 10 h (19/10/21 - 20/10/21)
    - Dependències: -

5.2. Tasca *Nodetool*

- **N – Temps dedicat a l'eina *Nodetool***
    - Descripció: Temps invertit en aprendre a utilitzar l'eina descartada *Nodetool* i en fer un *script* que l'utilitzés.
    - Durada: 30 h (21/10/21 - 06/01/22)
    - Dependències: -

5.3. Tasques *Sprint 1*

- **D1 – Disseny del codi per recollir dades de *Cassandra* amb *MBeans* i *JConsole*.**
    - Descripció: Disseny i plantejament del codi per a recollir dades de *Cassandra* mitjançant *MBeans* i *JConsole*.
    - Durada: 9 h (07/01/22 - 09/01/22)
    - Dependències: -

- **I1 – Implementació del codi per recollir dades de *Cassandra* amb *MBeans* i *JConsole*.**
    - Descripció: Implementació del codi per recollir dades de *Cassandra* amb *MBeans* i *JConsole*.
    - Durada: 48 h (10/01/22 - 25/01/22)
    - Dependències: D1

- **Y1 - Implementació d'un *YAML* i la seva lectura en el codi *Java*.**
    - Descripció: Implementació d'un *YAML* i la seva lectura en el codi *Java*.
    - Durada: 21 h (26/01/22 - 01/02/22)
    - Dependències: I1

- **T1 – *Testing* del codi per recollir dades de *Cassandra* amb *MBeans* i *JConsole*.**
    - Descripció: *Testing* del codi implementat per a recollir dades de *Cassandra* mitjançant *MBeans* i *JConsole*
    - Durada: 6 h (02/02/22 - 03/02/22)
    - Dependències: I1 i Y1

- **Doc1 – Documentació del codi per recollir dades de *Cassandra* amb *MBeans* i *JConsole*.**
    - Descripció: Documentació del procés de Disseny, Implementació i *testing* del codi per recollir dades de *Cassandra* amb *MBeans* i *JConsole*.
    - Durada: 9 h (04/02/22 - 06/02/22)
    - Dependències: T1

5.4. Tasques *Sprint 2*

- **D2 – Disseny dels gràfics a partir de les dades recollides**
    - Descripció: Disseny i plantejament d'un codi que creï gràfics a partir de les dades recollides a l'*Sprint 1.*
    - Durada: 9 h (07/02/22 - 09/02/22)
    - Dependències: I1
- **I2 – Implementació dels gràfics a partir de les dades recollides**
    - Descripció: Implementació del codi dissenyat que creï gràfics a partir de les dades recollides a l'*Sprint 1.*
    - Durada: 66 h (3/11/21 - 6/11/21)
    - Dependències: D2

- **T2 – *Testing* dels gràfics a partir de les dades recollides**
    - Descripció: *Testing* del codi implementat que creï gràfics a partir de les dades recollides a l'*Sprint 1.*
    - Durada: 6 h (04/03/22 - 05/03/22)
    - Dependències: I2

- **Doc2 – Documentació dels gràfics a partir de les dades recollides**
    - Descripció: Documentació del procés de Disseny, Implementació i *testing* del codi que creï gràfics a partir de les dades recollides a l'*Sprint 1.*
    - Durada: 9 h (06/03/22 - 08/03/22)
    - Dependències: T2

5.5. Tasques *Sprint 3*

- **D3 – Disseny de la interfície per visualitzar dades i gràfics**
    - Descripció: Disseny i plantejament d'una interfície per a poder visualitzar les dades i els gràfics obtinguts amb els *Sprints* 1 i 2.
    - Durada: 9 h (09/03/22 - 11/03/22)
    - Dependències: I2

- **I3 – Implementació de la interfície per visualitzar dades i gràfics**
    - Descripció: Implementació de la interfície dissenyada per a poder visualitzar les dades i els gràfics obtinguts amb els *Sprints* 1 i 2.
    - Durada: 66 h (12/03/22 - 02/04/22)
    - Dependències: D3

- **T3 – *Testing* de la interfície per visualitzar dades i gràfics**
    - Descripció: *Testing* de la interfície implementada per a poder visualitzar les dades i els gràfics obtinguts amb els *Sprints* 1 i 2.
    - Durada: 6 h (03/04/21 - 04/04/22)
    - Dependències: I3

- **Doc3 – Documentació de la interfície per visualitzar dades i gràfics**
    - Descripció: Documentació del procés de Disseny, Implementació i *testing* de la interfície implementada per a poder visualitzar les dades i els gràfics obtinguts amb els *Sprints* 1 i 2.
    - Durada: 9 h (05/04/22 - 07/04/22)
    - Dependències: T3

5.6. Tasques Finals

- **T4 – *Testing* general per arreglar *bugs* i problemes**
    - Descripció: *Testing* per revisar que tots els *sprints* funcionen correctament i que no hi ha cap *bug* que no s'hagués detectat.
    - Durada: 15 h (08/04/22 - 17/04/22)
    - Dependències: I3

- **Doc4 – Documentació final del projecte**
    - Descripció: Realització de la documentació final del projecte que contindrà totes les documentacions anteriors.
    - Durada: 15 h (08/04/22 - 17/04/22)
    - Dependències: Doc3

- **P – Presentació**
    - Descripció: Preparació de la presentació del projecte.
    - Durada: 21 h (18/04/22 - 26/04/22)
    - Dependències: Doc4

5.7. Tasques de Control

- **C1..27 – Reunions de Control**
    - Descripció: Reunions realitzades setmanalment amb els directors del projecte per revisar el que s'ha fet durant la setmana.
    - Durada: 27 h (21/09/22 – 08/02/22 tots els dimarts no festius de 12 a 13, 17/02/22 – 21/04/22 tots els dijous no festius de 12 a 13)
    - Dependències: -

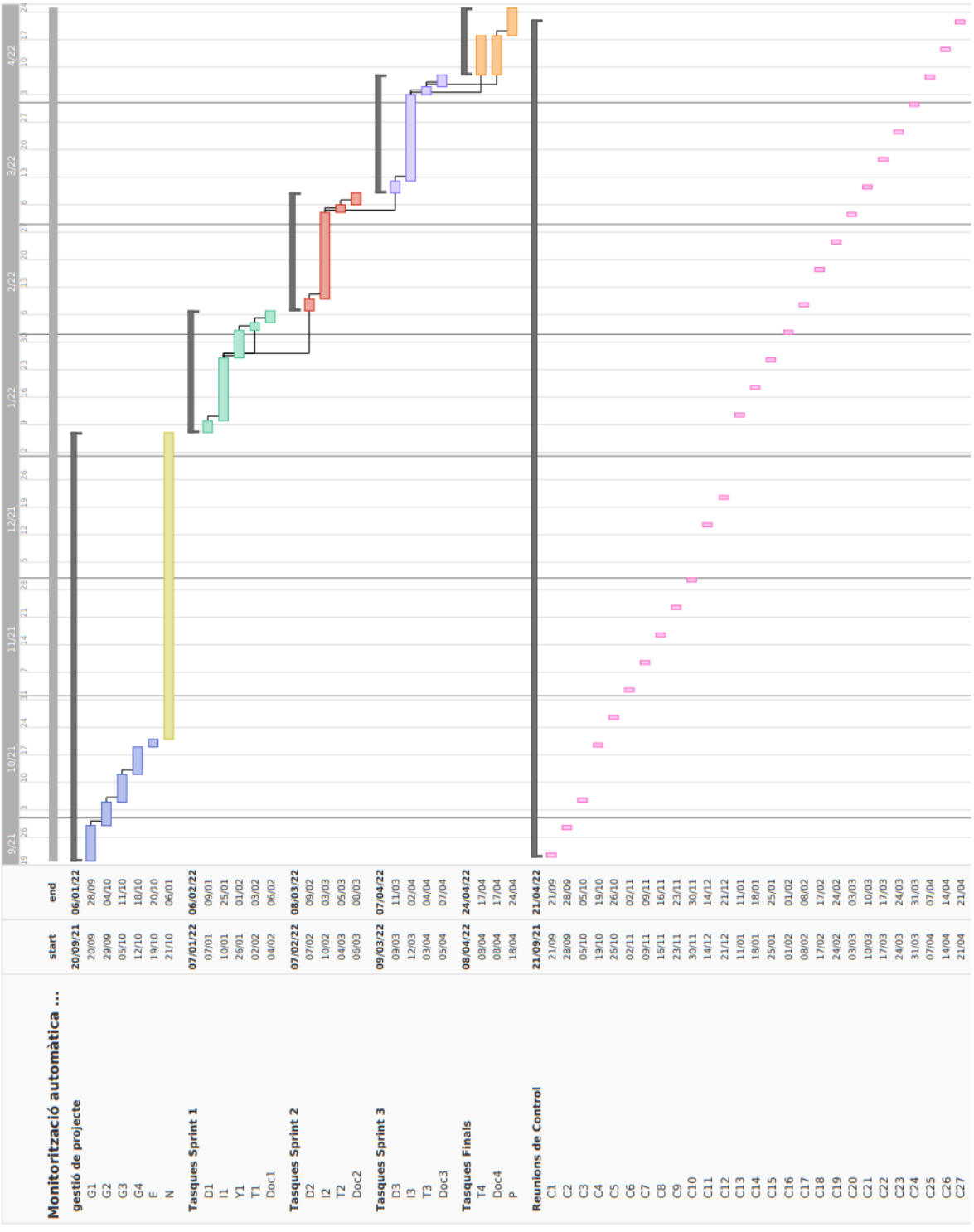
5.8. Taula resum de les tasques

ID	Tasca	Hores Pla. Ini	Hores Finals	Dependències
Gestió		**70**	**70**	
G1	Abast i contextualització del projecte.	25	25	
G2	Planificació temporal.	8	8	G1
G3	Gestió econòmica i sostenibilitat.	9	9	G2
G4	Document final.	18	18	G3
E	Estudi i anàlisi de l'estat de l'art.	10	10	
Nodetool		-	**30**	
N	Temps dedicat a l'eina *Nodetool*	-	-	
Sprint1		**55**	**93**	
D1	Disseny del codi per recollir dades de *Cassandra* amb *MBeans* i *JConsole*.	10	9	
I1	Implementació del codi per recollir dades de *Cassandra* amb *MBeans* i *JConsole*.	20	48	D1
Y1	Implementació d'un *YAML* i la seva lectura en el codi *Java*.	-	21	I1
T1	*Testing* del codi per recollir dades amb *MBeans* i *JConsole*	15	6	I1 i Y1
Doc1	Documentació del codi per recollir dades de *Cassandra* amb *MBeans* i *JConsole*.	10	9	T1
Sprint2		**55**	**90**	
D2	Disseny dels gràfics a partir de les dades recollides.	10	9	I1
I2	Implementació dels gràfics a partir de les dades recollides.	20	66	D2
T2	*Testing* dels gràfics a partir de les dades recollides.	15	6	I2
Doc2	Documentació dels gràfics a partir de les dades recollides.	10	9	T2
Sprint3		**55**	**90**	
D3	Disseny de la interfície per visualitzar dades i gràfics.	10	9	I2
I3	Implementació de la interfície per visualitzar dades i gràfics.	20	66	D3
T3	*Testing* de la interfície per visualitzar dades i gràfics.	15	6	I3
Doc3	Documentació de la interfície per visualitzar dades i gràfics.	10	9	T3
Tasques finals		**205**	**51**	
T4	*Testing* general per arreglar *bugs* i problemes.	87	15	I3
Doc4	Documentació final del projecte.	88	15	Doc3
P	Presentació.	30	21	Doc4
Control		**14**	**27**	
C1..27	Reunions de Control.	14	27	
TOTAL		**454**	**451**	

Taula 1: *Taula resum de tasques i hores*

6. Estimacions i Gantt

En la Imatge 2 que es troba a continuació es pot veure el diagrama de Gantt d'aquest projecte. El codi de colors d'aquest es correspon amb el codi de colors de la Taula 1. En tots els casos la responsable de les tasques soc jo, l'autora d'aquest projecte.



Imatge 1: *Diagrama de Gantt de la planificació del treball*

7. Pressupost

7.1. Identificació i estimació de costos

Qualsevol projecte suposa uns costos associats tant en recursos humans (és a dir, en el personal que du a terme el projecte), com en recursos materials (és a dir, en el *software* i *hardware* necessaris per al desenvolupament). Així doncs, a continuació es llisten totes les despeses i costos que s'estima que suposarà el desenvolupament d'aquest projecte.

7.1.1. Recursos humans

Els recursos humans són totes aquelles persones necessàries per dur a terme el projecte.

Tot i que aquest projecte només el desenvolupa una persona, s'ha decidit dividir les tasques en diferents rols per tal de simular així un equip real de desenvolupament i, per tant, estimar de forma més acurada els costos reals d'aquest projecte.

Per tal d'estimar els sous mitjans actuals de cada rol s'ha utilitzat la web Glassdoor. (8)

En la taula següent es troben els tres rols establerts amb el sou mitjà brut per hora i els sou amb la seguretat social inclosa que suposarà un 30% del sou brut.

ROL	Sou/h (Brut)	Sou/h + SS
Cap de projecte	20,12 €	26,16 €
Desenvolupador de *Software*	16,30 €	21,19 €
Tester	15,9 €	20,67 €

Taula 2: *Sou per hora de cada rol del projecte*

Un cop definit el sou per hora de cada rol es pot definir el cost total dels recursos humans el qual es pot veure reflectit en la següent taula. Per a fer-ho s'utilitzaran els codis de tasca definits en la planificació temporal. Tots els totals s'han calculat mitjançant la fórmula:

*Total = Hores dedicades * sou del rol*

Tasca	Temps de dedicació (h)			Total (€)
	Desenvolupador	Cap de projecte	Tester	
G1	-	25 h	-	654,0 €
G2	-	8 h	-	209,28 €
G3	-	9 h	-	235,44 €
G4	-	18 h	-	470,88 €
E	10 h	-	-	211,9 €
N	30 h	-	-	635,7 €
D1	9 h	-	-	190,71 €
I1	48 h	-	-	1.017,12 €
Y1	21 h	-	-	444,99 €
T1	-	-	6 h	124,02 €
Doc1	4 h	5 h	-	215,56 €
D2	9 h	-	-	190,71 €
I2	66 h	-	-	1.398,54 €
T2	-	-	6h	124,02 €
Doc2	4 h	5 h	-	215,56 €
D2	9 h	-	-	190,71 €
I3	66 h	-	-	1.398,54 €
T3	-	-	6 h	124,02 €
Doc3	4 h	5 h	-	215,56 €
T4	-	-	15 h	310,05 €
Doc4	7 h	8 h	-	357,61 €
P	21 h	-	-	444,99 €
C1..14	13 h	14 h	-	641,71 €
TOTAL	321 h	97 h	33 h	**10.021,62 €**

Taula 3: *Cost dels recursos humans per tasca*

Cal esmentar que aquesta part del pressupost ha patit una desviació a la baixa de 121,43 € envers el pressupost inicial a causa del canvi en la repartició d'hores de cada Rol i la disminució d'hores envers la planificació inicial.

7.1.2. Recursos Materials

A part dels recursos humans, també s'ha de comptabilitzar el cost tant del *software* com del *hardware* que s'utilitzarà al llarg del projecte.

En primer lloc, tenim el *software*. En el cas d'aquest projecte tot el *software* que s'utilitzarà és gratuït així que no suposarà cap cost addicional.

Pel que fa al *hardware*, tenim, per una banda, els recursos que s'utilitzaran per programar, que són, el portàtil des del qual es treballarà, el teclat i el ratolí, i,

per altra banda, *Minerva*, el *clúster* amb base de dades *Cassandra* a la que s'accedirà per fer el *testing* i part del desenvolupament.

En aquest cas, *Minerva* és propietat del DAC (Departament d'Arquitectura de computadors de la UPC) així doncs no es pot calcular un cost concret d'utilitzar-lo, ja que no es pot llogar. Així i tot, es farà un càlcul aproximat mitjançant el lloguer d'una base de dades *Cassandra* externa (9). Aquesta tindrà unes prestacions de 4vCores, 8GB de RAM i 60GB de Disc SSD.

Per altra banda, cal aclarir que dels recursos esmentats se'n calcula l'amortització al llarg del projecte que té una durada de set mesos. La part corresponent als set mesos és la que es comptabilitzarà en el pressupost final. Es calcula mitjançant la fórmula:

*Amortització = (Preu total * 7 mesos) / (anys de vida útil * 12)*

Així doncs, a la següent taula es veuen reflectits els costos de *hardware* del projecte.

Recurs	Preu (€)	Vida útil	Amortització (€)
Portàtil HP NoteBook 15-DA1009NS	499,00 €	4 anys	72,77 €
Teclat Mars Gaming	39,88 €	5 anys	4,65 €
Ratolí Mars Gaming	9,04 €	5 anys	1,05 €
BD *Cassandra*	21,10 €/mes	-	168,8 €
TOTAL	-	-	**247,27 €**

Taula 4*: Cost dels recursos hardware*

Aquesta part del pressupost ha patit una desviació a l'alça de 85,72 € envers el pressupost inicial a causa de l'augment de mesos de duració del treball (de 5 a 7).

7.1.3. Altres Despeses

A part de les despeses ja esmentades, en un projecte s'involucren altres factors que també suposen un cost, com serien l'electricitat, l'internet o la localització en la qual es treballarà.

Pel que fa a la localització, aquesta no es tindrà en compte, ja que es treballarà des de casa. Per altra banda, cal esmentar que el cost de l'electricitat actualment està disparat i que els càlculs es faran d'acord amb el preu habitual (10), tot i que és probable que aquest fluctuï molt al llarg del projecte.

A la següent taula es mostren les despeses d'electricitat i internet tenint en compte que un ordinador portàtil consumeix al voltant de 0,275 kWh.

Recurs	Preu (€)	Unitats	Amortització (€)
Electricitat	0.29 €/kWh	451 h	35,97 €
Internet	35 €/mes	8 mesos	280,0 €
Total	**-**	**-**	**315,97 €**

Taula 5: *Despeses addicionals del projecte*

Aquest apartat ha patit una desviació de 104,77 € a l'alça a causa dels canvis de planificació.

7.1.4. Contingències

A més a més de les despeses calculades fins ara, s'han de tenir en compte els costos de contingència. Aquests són els diners que s'afegeixen al pressupost per tal de tenir contemplats també possibles problemes que puguin sorgir al llarg del desenvolupament. Així doncs, tot i que en el càlcul d'hores destinats al projecte ja s'ha sigut generós per tal de cobrir possibles imprevistos, addicionalment s'afegirà al pressupost total un 10% extra.

7.1.5. Pressupost final

A continuació es mostra la suma total del pressupost del projecte:

Recurs	Preu(€)
Recursos Humans	10.021,62 €
Recursos Materials	247,27 €
Altres Despeses	315,97 €
SUBTOTAL	**10.584,86 €**
Contingències(10%)	1.058,5 €
TOTAL	**11.644,5 €**

Taula 6*: Pressupost final del projecte*

7.2. Control de gestió

Per tal de poder portar un control acurat i una correcta gestió de les desviacions que es puguin arribar a donar al llarg del desenvolupament del projecte s'ha anat actualitzant el pressupost al final de cada *sprint* d'acord amb les verdaderes hores emprades en cada tasca envers les hores que se li van estimar.

Per tal de portar aquest control s'utilitzaran les següents mètriques:

- **Desviació d'hores per tasca**

  Hores estimades – Hores reals

- **Desviació cost de recursos humans per tasca**

  Cost estimat de la tasca – Cost real de la tasca

- **Desviació cost de recursos materials**

  Cost estimat del material – cost real del material

- **Desviació altres despeses**

  Cost estimat d'altres despeses – Cost real d'altres despeses

- **Desviació total d'hores**

  Hores totals estimades – Hores reals estimades

- **Desviació total de costos**

  Cost total estimat – Cost total real

8. Sostenibilitat

Després de realitzar l'enquesta de sostenibilitat he pogut extreure diverses conclusions i pensaments.

En primer lloc, crec que tinc una idea general bastant amplia del que és la sostenibilitat envers les TIC, sobretot gràcies a les assignatures cursades al llarg de la carrera, però també gràcies a principis propis que em formen com a persona.

Tot i així, i tot i tenir una visió general bastant bona del que suposa la sostenibilitat en un camp com és la tecnologia i la informàtica, considero que de les tres branques que la componen (ambiental, econòmica i social), la branca econòmica em flaqueja una mica, probablement perquè mai m'he enfrontat a un projecte real en una empresa en la qual els costos del projecte suposin un impediment.

A més, pel que fa als altres dos camps, crec que em falten recursos i sobretot experiència per a poder aplicar millores als projectes que desenvolupo per a fer-los més sostenibles i em falten coneixements de quins indicadors existeixen i de com puc usar-los.

De totes maneres, i per acabar, estic bastant satisfeta amb els meus coneixements de sostenibilitat, ja que en crear un projecte sempre intento tenir en compte que contamini el mínim, que el projecte realment sigui útil i que ho sigui durant un període de temps prolongat, que els costos de producció siguin el màxim de reduïts i que no suposi un increment de la desigualtat social sinó tot el contrari, que si és possible, ajudi a reduir la bretxa tecnològica.

8.1. Dimensió ambiental

- **Has estimat l'impacte mediambiental que tindrà la realització del projecte? T'has plantejat minimitzar l'impacte, per exemple, reutilitzant recursos?**

  Com que el projecte és una solució *software* i no *hardware* no tindrà un gran impacte ambiental més enllà del que suposi l'ús dels dispositius pel desenvolupament.

  Per tal de minimitzar l'impacte com a molt es podria intentar utilitzar energies renovables per a alimentar el portàtil des del qual es desenvoluparà o bé la base de dades que s'utilitzarà per al *testing*.

- **En què millorarà ambientalment la teva solució a les existents?**

  El fet que el projecte sigui de codi obert el farà més accessible i flexible i, per tant, es podrà adaptar millor a la màquina en la qual s'utilitzi perquè consumeixi el mínim.

  A més el projecte com a tal ajuda a veure dades de rendiment de la base de dades i, per tant, rectificar-ne els problemes, amb el qual es pot aconseguir un consum operacional menor i una optimització dels recursos.

8.2. Dimensió econòmica

- **Has estimat el cost de la realització del projecte (recursos humans i materials)?**

  S'ha elaborat un pressupost amb tots els recursos (humans i materials) per a poder estimar el cost econòmic del projecte.

- **En què millorarà econòmicament la teva solució a les existents?**

  El projecte, com ja s'ha esmentat anteriorment, és de codi obert, així doncs, és gratuït i, per tant, totes aquelles empreses que actualment utilitzin una solució privada es poden estalviar la inversió.

  A més el fet d'optimitzar l'ús dels recursos gràcies a les dades extretes de l'aplicació creada al llarg del projecte fa que el consum energètic es redueixi i, per tant, que es redueixi el cost econòmic per operació.

8.3. Dimensió social

- **Què creus que t'aportarà en l'àmbit personal la realització d'aquest projecte?**

  En l'àmbit personal aquest projecte m'aportarà molta experiència i coneixements, tant en l'àmbit tècnic com en l'àmbit de gestió, que em podran ser útils en futurs projectes.

- **En què millorarà socialment (qualitat de vida) la teva solució a les existents?**

  L'eina que es desenvoluparà al llarg del projecte suposarà un estalvi de temps als gestors i usuaris de la base de dades, gràcies al fet que l'extracció de dades de rendiment serà molt més ràpida i agradable.
  Així doncs, aquest temps guanyat el podran invertir en altres menesters de més profit.

- **Existeix una necessitat real del projecte?**

  És cert que ja hi ha solucions al mercat a aquest problema concret, però totes elles són de pagament. Així doncs, realment el meu projecte si aporta algun benefici, no només personal, sinó també a altres usuaris.

9. Marc Teòric

Aquest apartat pretén introduir breument conceptes teòrics sobre el funcionament intern de *Cassandra* i les mètriques que s'han decidit extreure, així com què són els *MBeans* que s'han utilitzat per agafar les dades i què és un *YAML*, tot això per tal d'entendre millor el funcionament de l'aplicació, la seva implementació i el perquè de la selecció de paràmetres que s'ha fet.

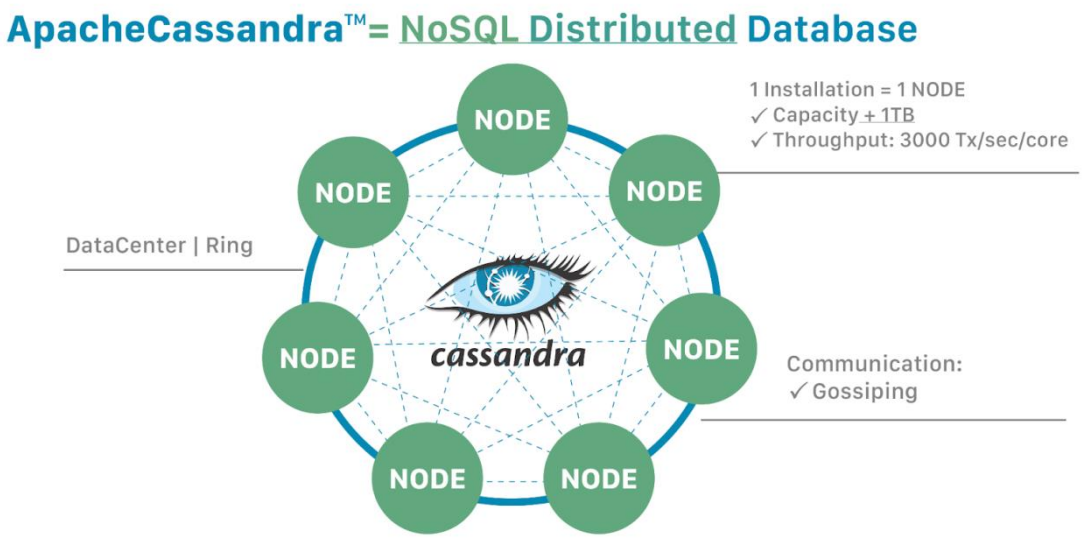
9.1. Arquitectura de *Cassandra*

Per a poder entendre adequadament les dades i estadístiques que extreu l'eina que desenvolupa aquest treball, cal primer entendre l'arquitectura de les bases de dades *Cassandra*, per tal de poder veure perquè són importants tals estadístiques i a més poder valorar si la base de dades funciona adequadament d'acord amb aquestes. Així doncs, a continuació s'expliquen breument els conceptes clau de l'arquitectura de *Cassandra*.

La informació d'aquest apartat s'ha extret íntegrament de la referència (11)

En primer lloc, com ja s'ha contat en apartats anteriors, *Cassandra* és una base de dades de codi obert, distribuïda i *NoSQL* que està pensada per tractar amb grans quantitats de dades. Partint d'això a continuació es defineixen més característiques de l'arquitectura de les bases de dades *Cassandra*.

Cassandra funciona mitjançant múltiples nodes interconnectats entre si que treballen junts formant un *ring* o *data center* i que segueixen una arquitectura *peer-to-peer*, és a dir, que no hi ha un node central que controla els altres, sinó que tots estan al mateix nivell. Això la fa resistent a fallades, ja que si cau un node, els altres es distribueixen la feina per a continuar funcionant.



Imatge 2: *Esquema de l'arquitectura de Cassandra* **(24)**

La informació que es guarda a la base de dades es distribueix de forma equitativa entre tots els nodes que formen el *data center*, en porcions (identificades per *primary keys*) que són l'equivalent a una fila d'una taula (és a dir que és una base de dades *row-oriented*) i a més es distribueix de forma automàtica.

Addicionalment, es pot afegir replicació per tal que les dades es trobin repetides en més d'un node i així les lectures siguin més ràpides i es guanyi fiabilitat envers fallades de nodes.

Quan un usuari es connecta a un node concret, aquest treballa com a coordinador i envia les peticions de lectura o escriptura als altres nodes. En el cas de les escriptures, aquestes s'emmagatzemen en el que anomenem *Memtable* que funciona com una *cache*, quan aquesta s'omple la informació es volca a disc en una nova *SSTable*. Les *SSTables* no se sobreescriuen, sinó que cada cop que es volca una *Memtable* se'n crea una de nova i, per tant, es repeteixen les files però amb modificacions. Per indicar que una fila està desactualitzada aquesta es marca i la base de dades periòdicament consolida les *SSTables* per eliminar informació antiga o redundant.

Per a mantenir la coherència entre tots els nodes del *data center*, cada un d'aquests es comunica i s'envia informació del seu estat actual mitjançant un protocol *gossip* que s'enregistra en un *commit log*, el qual en cas de fallada permet recuperar les escriptures que s'haguessin pogut perdre.

Quant a la lectura, primer es consulta la *Memtable*, si no es troba, es consulta la *cache* de files, en cas de no trobar-se allà la consulta desitjada s'accedeix a les *SSTable*, a les quals es busca la informació utilitzant una *cache* de claus que permet saber en quina *SSTable* està la informació més actualitzada.

9.2. Paràmetres a Monitoritzar

Un cop explicada l'arquitectura de *Cassandra* i les seves diferents parts més rellevants pel que fa a aquest treball, cal explicar els diferents paràmetres que s'han decidit monitorejar amb l'aplicació i el perquè s'han escollit, així doncs a la següent llista podeu trobar tots els paràmetres que es poden consultar amb l'aplicació i els seus diferents atributs:

1 – AllMemtablesHeapSize

- Value

Com el seu nom bé indica, aquest paràmetre ens diu quin espai de memòria ocupen totes les *Memtables*. Com s'ha pogut veure a l'apartat anterior, les *Memtables* son part fonamental del funcionament de *Cassandra*, així doncs és interessant saber quant ocupen en total, per poder-nos fer una idea de si s'està gestionant bé el traspàs de *Memtable* a *SSTable*.

2 - CompletedTasks

- Value

Per poder llegir bé la resta de paràmetres cal saber quantes tasques s'han completat en total en el node, i això és el que ens indica aquest paràmetre.

3 - Exceptions

- Count

Les excepcions solen ser una mala senyal, és per això que és adequat vigilar-les i veure quantes s'han produït en total, ja que si són moltes pot ser un indicador que alguna cosa no va bé.

Els paràmetres del 4 al 7 donen informació sobre la *Key Cache*, que com ja s'ha explicat a l'apartat anterior és el tercer lloc al qual s'accedeix quan es fa una lectura, en cas que no s'hagi trobat la informació a la *Memtable* o la *Row Cache*, és per això que obtenir-ne informació val la pena, ja que si s'hi fan moltes peticions, però hi ha molts errors o pocs encerts és possible que la feina no s'estigui repartint correctament entre nodes o en un ordre òptim.

4 - KeyCacheHitRate

- Value

5 - KeyCacheHits

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

Tant el paràmetre 4 com 5 donen informació sobre els encerts en els accessos a la *Key Cache*.

6 - KeyCacheMisses

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

Aquest paràmetre dona informació sobre les fallades en els accessos a la *Key Cache*.

7 - KeyCacheRequests

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

I aquest dona informació sobre els accessos totals a la *Key Cache*, que hi hagi molts accessos en total, també pot ser un mal indici, ja que aquest és el tercer nivell de memòria al que s'accedeix, i si tot funcionés adequadament, la majoria de consultes s'haurien de poder suplir amb els altres dos nivells (*Memtables* i *Row Cache*).

Els següents 4 paràmetres (del 8 a l'11) ofereixen informació sobre les lectures fetes al node que s'està monitoritzant. Poder veure com es comporten les lectures pot ajudar a veure si els diferents nivells de memòria funcionen correctament, i en cas de monitoritzar més d'un node a la vegada, pot ajudar a veure si la càrrega de treball es reparteix de forma equitativa entre els nodes.

8 - ReadLatency

- 50thPercentile
- 75thPercentile
- 95thPercentile
- 98thPercentile
- 999thPercentile
- 99thPercentile
- Count
- FifteenMinuteRate
- FiveMinuteRate
- Max
- Mean
- MeanRate
- Min
- OneMinuteRate
- StdDev

Aquest paràmetre dona informació sobre la latència de lectura, és a dir que dona informació sobre el temps que transcorre d'ençà que es fa una petició de lectura a la memòria fins que aquesta la supleix.

9 - ReadTimeOuts

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

Els *Time Outs* són aquelles peticions que tarden més de lo esperat en servir-se, així doncs tenir molts *Time Outs* és una mala senyal.

10 - ReadTotalLatency

- Count

Aquest paràmetre complementa el paràmetre 8, donant una latència total de lectura, envers el desglossament de l'altre paràmetre.

11 - ReadUnavailables

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

Els *Unavailables*, com indica el seu nom són lectures que no estan disponibles, que no s'han pogut suplir ni tan sols fora del temps esperat. Aquestes també suposen un problema i són un indicador de mal funcionament.

Així com els paràmetres del 4 al 7 eren indicadors de les *Key Cache*, els següents 4 paràmetres (del 12 al 15) són indicadors de la *Row Cache*, que és el segon nivell de memòria després de les *Memtables*. És per això que aquests són necessaris per al mateix motiu que els paràmetres de la *Key Cache*.

12 - RowCacheHitRate

- Count

13 - RowCacheHits

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

Tant el paràmetre 12 com 13 donen informació sobre els encerts en els accessos a la *Row Cache*.

14 - RowCacheMisses

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

Aquest paràmetre dona informació sobre les fallades en els accessos a la *Row Cache*.

15 - RowCacheRequests

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

I aquest dona informació sobre els accessos totals a la *Row Cache*, igual que a la *Key Cache* que hi hagi molts accessos en total, també pot ser un mal indici tot i que hi hauria d'haver més accessos en aquesta que en la *Key Cache*, ja que tots els accessos a la *Key* passen primer per la *Row*.

16 - TotalDiskSpaceUsed

- Value

Aquest paràmetre dona informació sobre l'espai total de disc utilitzat, aquesta informació pot ser útil per veure si el sistema s'està desfent correctament de les *SSTables* desfasades o si està acumulant sense parar.

Finalment, equivalentment a les Lectures, tenim els paràmetres de les Escriptures, que un cop més poden ser útils per veure si les peticions d'escriptura se supleixen prou ràpidament o si s'estan perdent escriptures, coses que també poden indicar un mal funcionament del sistema.

17 - WriteLatency

- 50thPercentile
- 75thPercentile
- 95thPercentile
- 98thPercentile
- 999thPercentile
- 99thPercentile
- Count
- FifteenMinuteRate
- FiveMinuteRate
- Max
- Mean
- MeanRate
- Min
- OneMinuteRate
- StdDev

Així com amb la latència de les lectures, aquest paràmetre ens dona informació sobre la latència de les escriptures, és a dir, informació sobre el temps que transcorre d'ençà que es fa una petició d'escriptura fins que aquesta es du a terme.

18 - WriteTimeOuts

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

Aquest paràmetre ens indica quantes escriptures han tardat més de l'esperat en realitzar-se.

19 - WriteTotalLatency

- Count

Aquest paràmetre complementa a l'altre paràmetre de latència d'escriptura (el número 17).

20 - WriteUnavailables

- Count
- FifteenMinuteRate
- FiveMinuteRate
- MeanRate
- OneMinuteRate

Per acabar, aquest paràmetre ens dona informació sobre les escriptures que s'han perdut, és a dir, les que no s'han pogut dur a terme.

9.3. Teoria *MBeans*

Per accedir als paràmetres anteriors i agafar-ne els valors s'ha fet ús d'una tecnologia de *Java* que ja venia integrada a *Cassandra* anomenada *MBeans*, aquests són el diminutiu de *Managed Beans* i formen part de *JMX* (*Java Management Extension*).

Els *MBeans* són un objecte *Java* que va relacionat amb algun component, aplicació o dispositiu de la computadora i el sistema (n'és la seva representació virtual), i en permeten la seva gestió i administració en remot oferint atributs i operacions que en faciliten la seva manipulació dins un codi *software*.

Tot i que els *MBeans* es poden crear des de zero per suplir qualsevol necessitat, i tot i que també es poden utilitzar per fer canvis en el component al qual estan vinculats mitjançant els mètodes que *l'MBean* inclogui, en el cas d'aquest treball s'han utilitzat els *MBeans* que ja existien a *Cassandra* i només s'han utilitzat per fer-hi consultes, no per fer-hi modificacions. (12) (13)

9.4. Teoria *YAML*

Finalment, perquè l'usuari pugui escollir quins paràmetres, d'entre la llista anterior, vol visualitzar, s'ha utilitzat un fitxer *YAML*, del qual se n'explicarà l'estructura en l'apartat 10.1., però primer, cal saber què és un *YAML*.

Un fitxer *YAML* és un fitxer extern que, mitjançant un llenguatge amigable i llegible per l'home, li passa informació a l'aplicació, habitualment per configurar-la.

El llenguatge *YAML* és compatible amb qualsevol llenguatge de programació, el que el fa molt portable i flexible. (14) (15)

10. La Meva Proposta

Aquest apartat pretén contar amb detall totes les funcionalitats de l'aplicació desenvolupada amb aquest treball, per a fer-ho es dividirà l'explicació primer en el *YAML* de configuració i després en l'aplicació com a tal. Els detalls de la implementació interna s'explicaran en el següent apartat.

Per a facilitar l'explicació s'inclouran captures de pantalla.

10.1. Resum general del funcionament

En trets generals l'aplicació el que fa es llegir un fitxer de configuració que li indicarà a quin node connectar-se i quins paràmetres i atributs consultar d'entre un conjunt preseleccionat, tot seguit crea una connexió amb la base de dades *Cassandra* (en el cas d'aquest treball a *Minerva*) i n'obté els valors dels paràmetres i atributs que s'han indicat al fitxer de configuració fent ús dels *MBeans*.

A continuació arranca una interfície gràfica, que és la que podrà manipular l'usuari. Aquesta mitjançant botons i pestanyes li ofereix a l'usuari la possibilitat de visualitzar gràfiques on es veu l'evolució dels valors de cadascun dels paràmetres, consultar-ne el valor numèric concret o guardar la gràfica/gràfiques i/o els valors numèrics que han anat prenent els atributs (tots els arxius desats amb un *time stamp*) per a poder consultar en un altre moment els resultats i comparar-los entre si.

10.2. Funcionament *YAML*

Com ja s'ha comentat anteriorment, l'aplicació disposa d'un fitxer *YAML* que permet a l'usuari escollir quin node, quins paràmetres i quins atributs de cada paràmetre vol monitoritzar.

Es va decidir utilitzar *YAML* com a format del fitxer de configuració per a seguir amb el criteri d'aplicació senzilla d'utilitzar per a l'usuari, ja que *YAML* es un format molt llegible, que es pot entendré només d'una ullada i que, per tant, a l'usuari li resultarà fàcil modificar.

El *YAML* té escrites totes les opcions en un format concret compatible amb l'aplicació que el llegeix i l'usuari només hi ha de descomentar aquelles opcions que vulgui. En les següents imatges (imatges 3 i 4) es pot veure part del *YAML*.

```yaml
---
#Escollir un dels dos nodes per monitorejar, no es poden escollir els dos a la vegada
nodos:
  - "Minerva-11"
  #- "Minerva-12"
```

Imatge 3: *Selecció de node al YAML*

Com es pot observar, en primer lloc, cal seleccionar el node de *Cassandra* que es vol monitoritzar. En aquest cas concret, i com ja s'ha esmentat, s'utilitza *Minerva* i per desenvolupar aquest treball s'ha tingut accés únicament als nodes 11 i 12 d'aquesta, així que al *YAML* només hi apareixen aquests dos.

Tot i això, sempre que es respecti el format, aquests nodes són modificables.

```yaml
parametros:
#Escollir quins paràmetres monitorejar d'entre els següents i escollir els
#atributs concrets a monitorejar: (treure el # per escollir-los)

#Si s'escull el parametre X s'ha de treure el # com a minim del
#name, num, atributs i almenys el d'un atribut

  - name: AllMemtablesHeapSize
    num: 1
    atributs:
    - Value

  #- name: CompletedTasks
    #num: 2
    #atributs:
    #- Value

  #- name: Exceptions
    #num: 3
    #atributs:
    #- Count

  #- name: KeyCacheHitRate
    #num: 4
    #atributs:
    #- Value

  - name: KeyCacheHits
    num: 5
    atributs:
    - Count
    - FifteenMinuteRate
    #- FiveMinuteRate
    - MeanRate
    - OneMinuteRate

  - name: KeyCacheMisses
    num: 6
    atributs:
    - Count
    - FifteenMinuteRate
    - FiveMinuteRate
    - MeanRate
    - OneMinuteRate

  #- name: KeyCacheRequests
    #num: 7
    #atributs:
    #- Count
    #- FifteenMinuteRate
    #- FiveMinuteRate
    #- MeanRate
    #- OneMinuteRate

  - name: ReadLatency
    num: 8
    atributs:
    - 50thPercentile
```

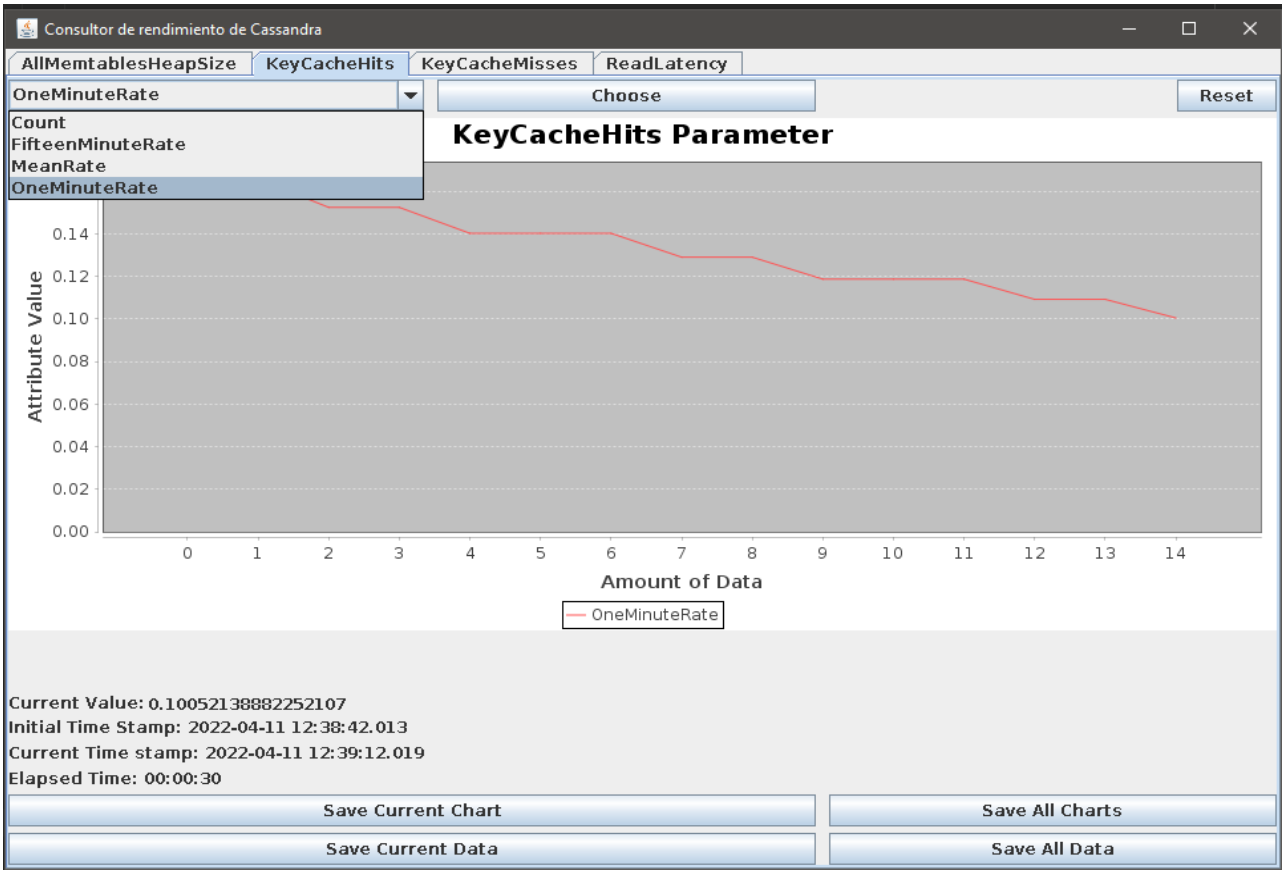
Imatge 4: *Part del llistat de paràmetres del YAML*

Per altra banda, veiem que l'usuari disposa d'un llistat dels paràmetres disponibles i els seus respectius atributs. En el cas de la imatge 4, veiem que hi ha quatre paràmetres descomentats, és a dir, seleccionats i uns altres quatre comentats i que alguns dels descomentats no tenen descomentats tots els atributs. Cal destacar que la imatge anterior és només una porció del *YAML*, i que en aquest hi ha el llistat sencer de paràmetres que s'han explicat en l'apartat 9.2.

En el cas dels paràmetres, a diferència dels nodes, actualment no se'n poden afegir extres de forma senzilla (caldria, com es veurà en l'apartat d'implementació, crear una nova classe per al nou paràmetre i modificar diverses parts del codi), així doncs l'usuari s'ha de limitar a comentar o descomentar la llista que ja hi ha, vigilant de mantenir el format.

10.3. Funcionalitats de l'Aplicació

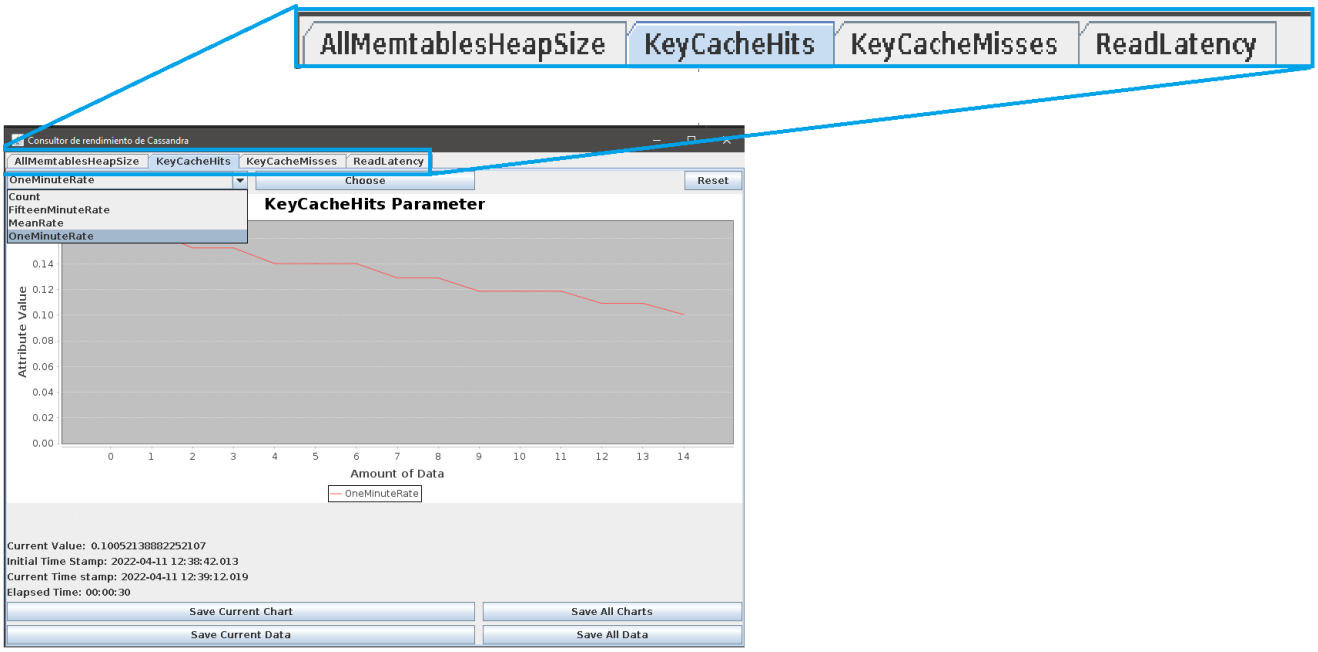
Un cop explicat el *YAML* passem a l'explicació de les funcionalitats de l'aplicació, la qual es pot observar en la següent imatge.



Imatge 5: *Captura general de la interfície gràfica del projecte*

Així doncs, en els següents punts s'explicarà què fa cadascun dels components i quines possibilitats ofereix l'aplicació. Totes les imatges que s'utilitzaran en aquest apartat són captures de l'aplicació configurada amb el *YAML* tal com s'ha mostrat en l'apartat anterior a la imatge 4.

- **Pestanyes:**



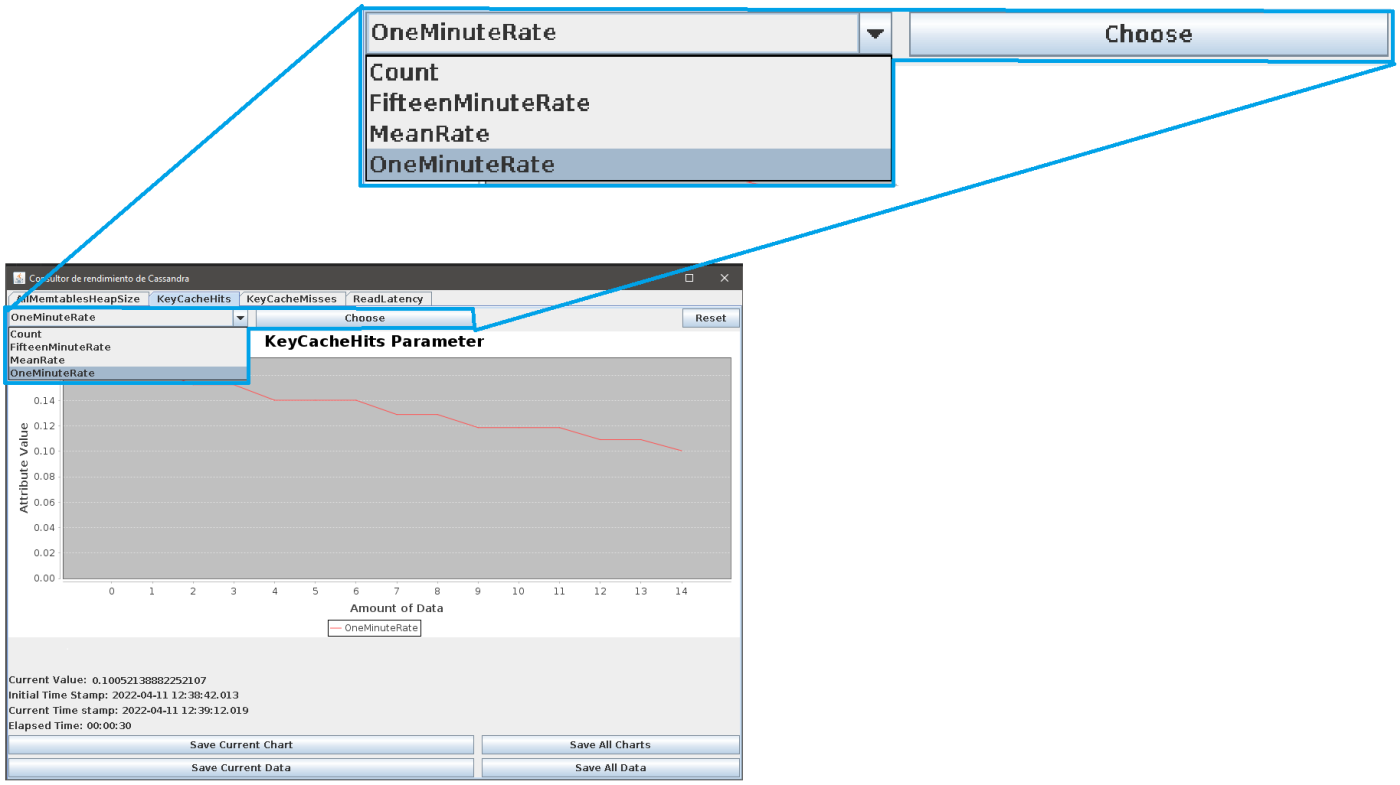
Imatge 6: *Captura de la interfície amb zoom a les pestanyes*

A la part superior de la finestra podem veure unes pestanyes que fan referència als paràmetres que s'estan monitoritzant. A l'aplicació apareixeran únicament aquells paràmetres que s'hagin seleccionat al *YAML*, fent així que la interfície quedi neta i clara, el que fa que l'usuari pugui trobar amb facilitat el que busca.

En seleccionar una pestanya se seleccionarà el paràmetre corresponent i es podran visualitzar si es desitja els seus atributs tal com s'explica més endavant.

A la imatge 6 es poden veure ampliades les pestanyes a les quals es fa referència.

- **Atributs:**



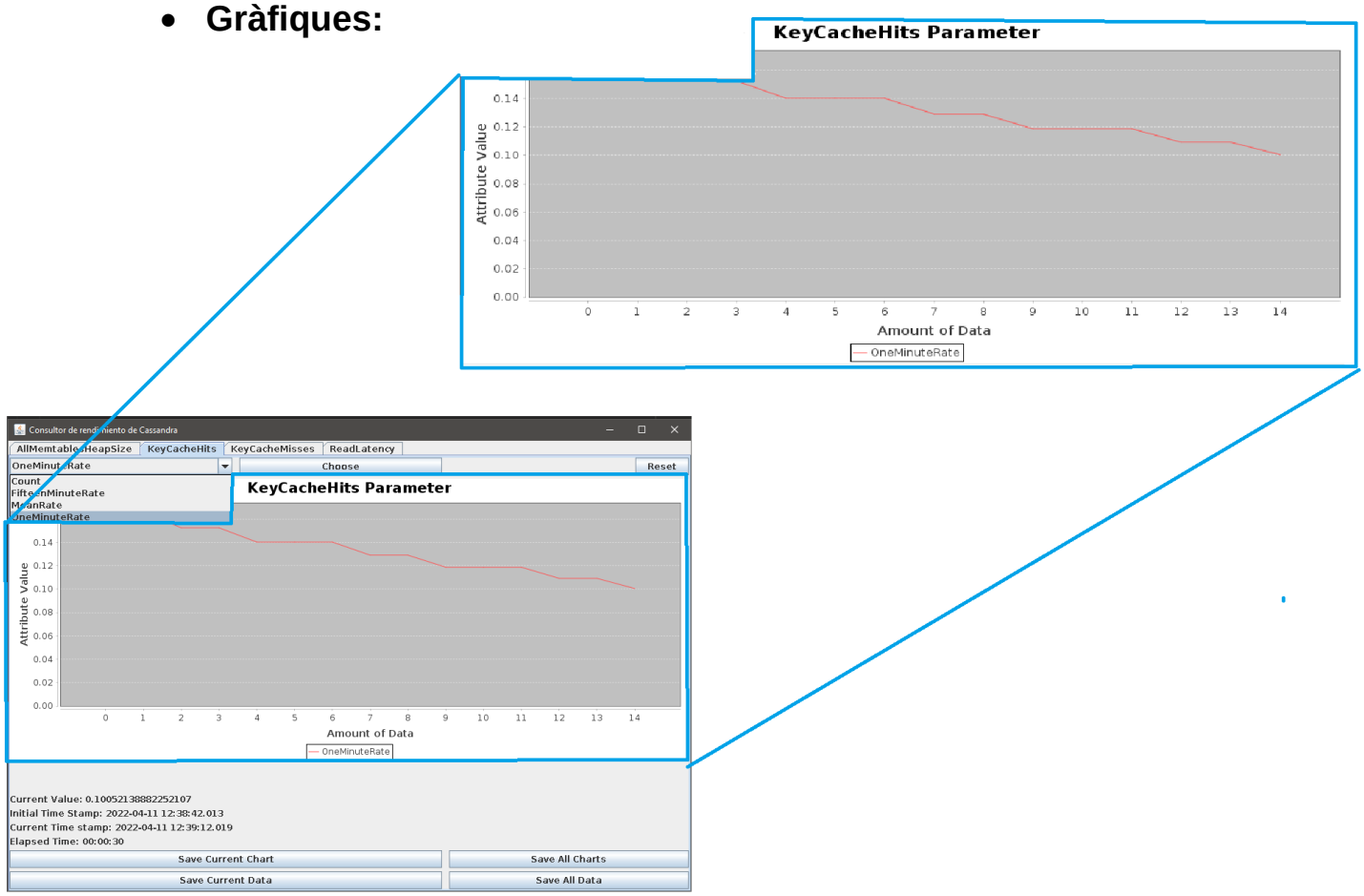
Imatge 7: *Captura de la interfície amb zoom a la selecció d'atributs*

Un cop seleccionat el paràmetre veiem que hi ha un desplegable en el qual es mostren els atributs que es poden visualitzar, un cop més, només hi apareixen aquells que han estat seleccionats al *YAML*, per tal de facilitar la cerca de l'usuari i evitar soroll innecessari.

Un cop seleccionat l'atribut concret que es vol visualitzar s'ha de prémer el botó *Choose*, el qual farà que es mostri la gràfica de l'atribut, amb els valors obtinguts d'ençà que s'ha arrancat l'aplicació.

A la imatge 7 es pot veure el desplegable i el botó *Choose* que s'esmenten.

- **Gràfiques:**



Imatge 8: *Captura de la interfície amb zoom a la gràfica*

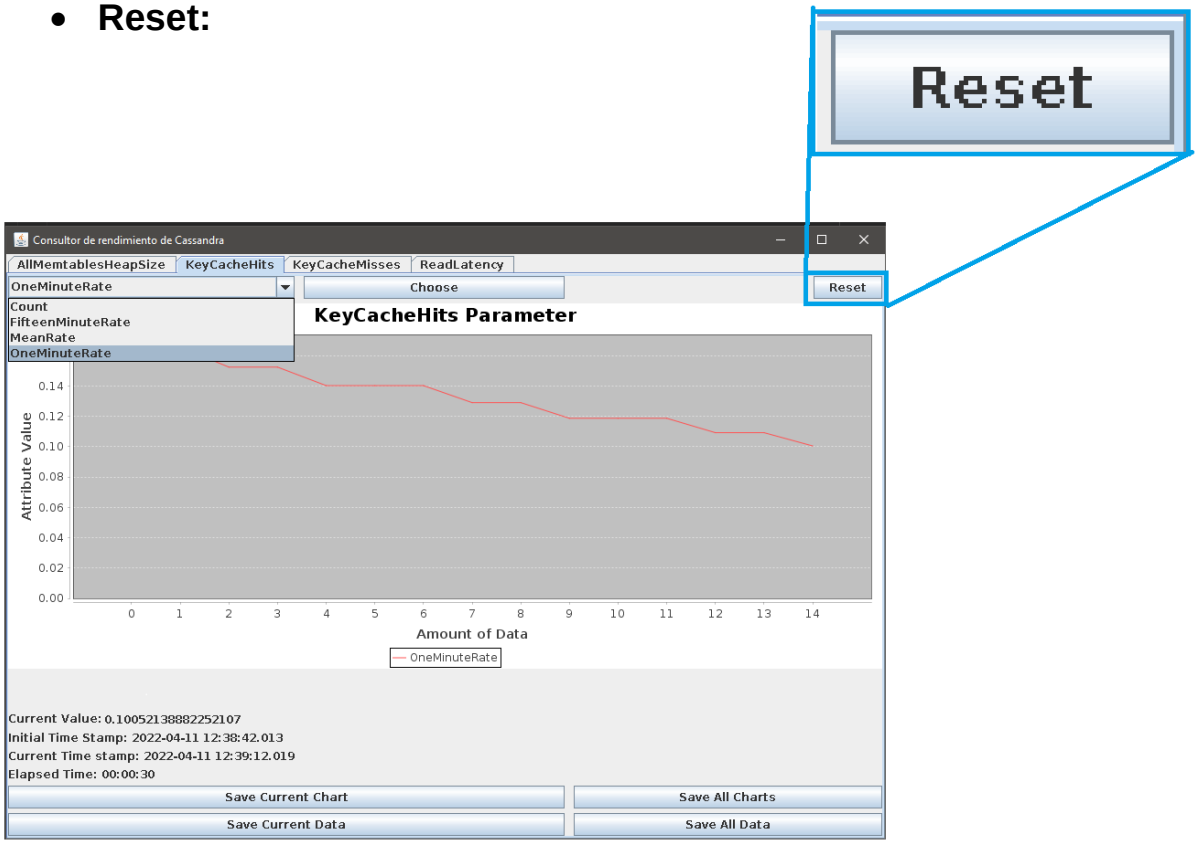
Un cop seleccionat l'atribut apareixerà la gràfica al centre de l'aplicació, tal com es veu en la imatge 8.

La gràfica mostra l'evolució del valor que pren l'atribut a mesura que transcorre el temps. L'eix Y n'indica el seu valor, i l'eix X la quantitat de valors presos. El títol de la gràfica indica quin paràmetre s'està monitoritzant i sota l'etiqueta de l'eix X, envoltat amb un requadre podem veure quin és l'atribut que representa la gràfica.

Un cop seleccionat l'atribut i creada la gràfica aquesta es mantindrà i continuarà afegint dades tot i canviar d'atribut o fins i tot de paràmetre, fent així que es pugui tornar a consultar-la en qualsevol moment.

Com ja s'ha esmentat anteriorment, les dades es comencen a obtenir en quant s'arrenca l'aplicació tot i no seleccionar la gràfica per visualitzar-la.

- **Reset:**



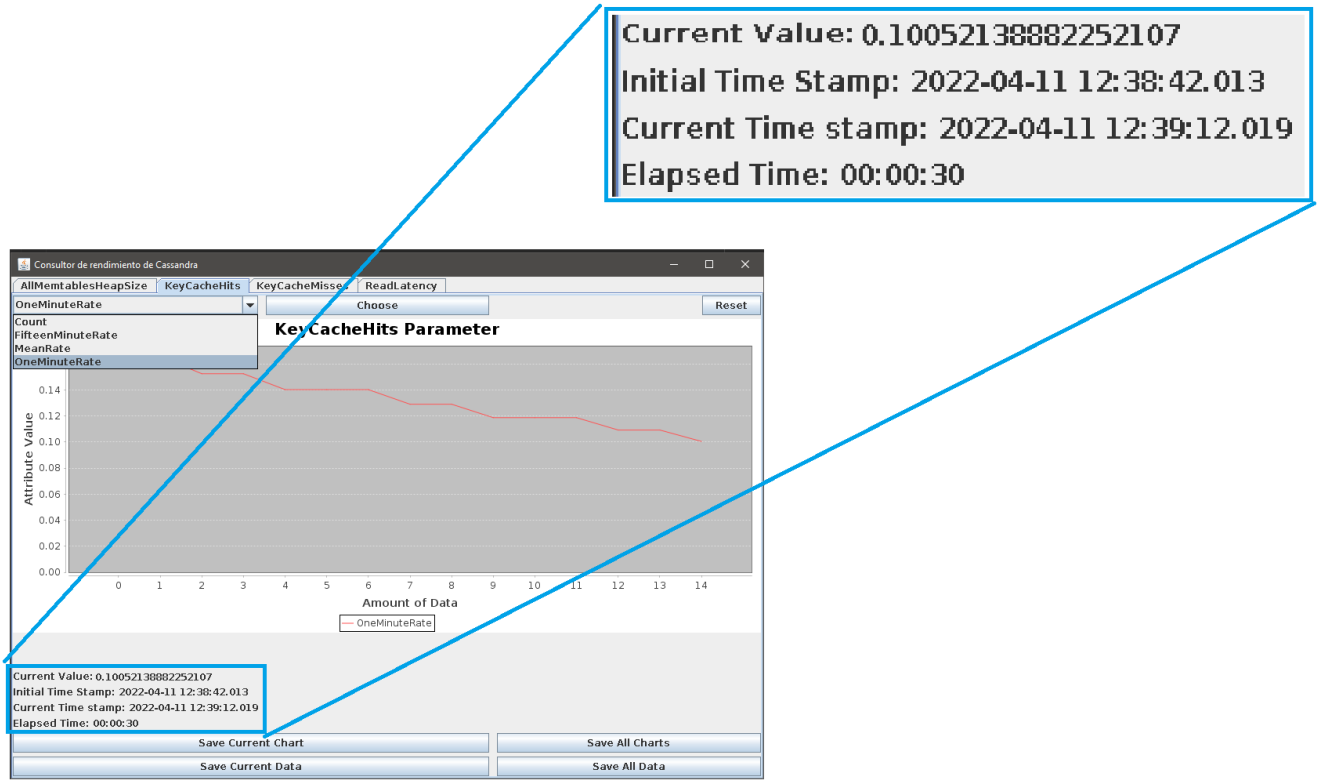
Imatge 9: *Captura de la interfície amb zoom al botó de reset*

Com que les gràfiques no tenen *scroll* lateral les dades recollides cada cop es veuen més ”apretades” i pot ser confús visualitzar-les, és per això que l’aplicació disposa d’un botó de *Reset* que reinicia la recollida de dades des del punt d’execució en el que es pitja.

Aquest botó només afecta la gràfica que s’estigui visualitzant en aquell moment.

A la imatge 9 podem veure on està ubicat.

- **Etiquetes:**



Imatge 10: *Captura de la interfície amb zoom a les etiquetes*

Per altra banda i com a informació addicional l'aplicació disposa de 4 etiquetes que li indiquen a l'usuari:

- El valor concret de l'atribut que s'està mostrant a la gràfica en el moment actual.

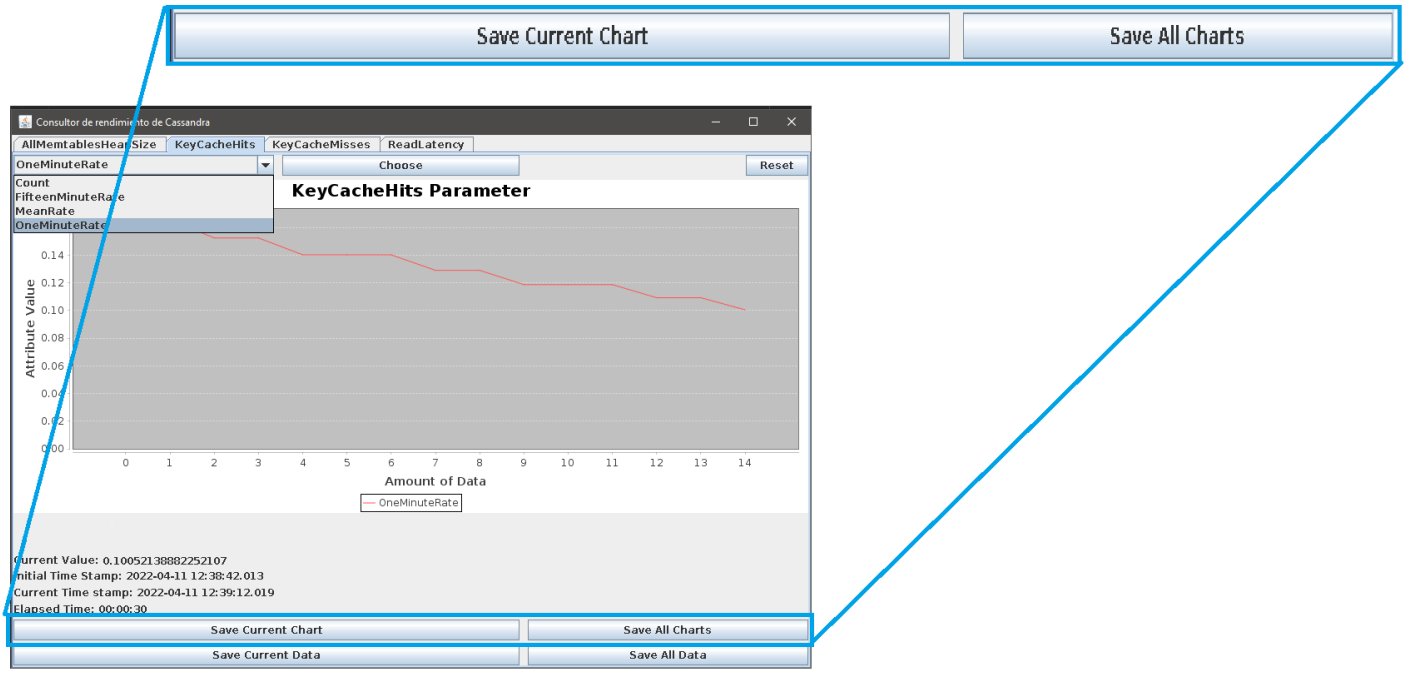
- El moment de temps en què ha arrancat l'aplicació.

- El moment de temps actual.

- Un cronòmetre amb la quantitat de temps transcorregut d'ençà que s'ha iniciat l'aplicació fins que el moment actual.

A la imatge 10 podem veure'n un exemple ampliat.

- **Guardat de Gràfiques:**



Imatge 11: *Captura de la interfície amb zoom als botons de Save Charts*

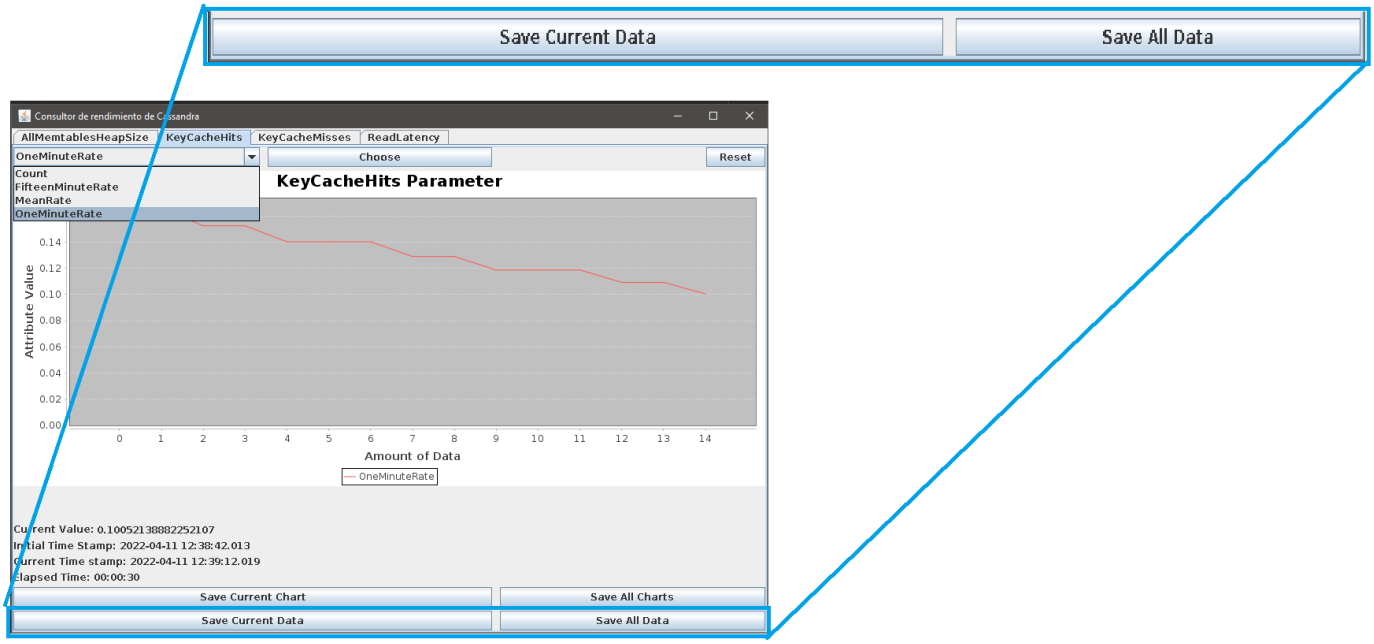
En cas que l'usuari vulgui guardar les gràfiques per consultar-les després, disposa de dos botons per a fer-ho (veure imatge 11).

El botó *Save Current Chart* guarda a la carpeta on s'ha executat l'aplicació i en format *.png* la gràfica que s'està visualitzant en aquell moment. Aquest *.png* pren el nom del paràmetre, l'atribut i un *time stamp*.

El botó *Save All Charts* guarda a la carpeta on s'ha executat l'aplicació i en format *.png* totes les gràfiques que s'hagin visualitzat per part de l'usuari d'ençà que s'ha arrancat l'aplicació (és a dir, que se n'hagi fet "*Choose*"). Totes amb els valors del moment en què es prem el botó, és a dir que totes reflecteixen el mateix instant de temps. En aquest cas els *.png* prenen el nom del paràmetre, un identificador de gràfica i un *time stamp.*

Això li permet a l'usuari poder comparar-les en el futur entre elles.

- **Guardat de Dades**



Imatge 12*: Captura de la interfície amb zoom als botons de Save Data*

Finalment, l'aplicació disposà dels botons *Save Current Data* i *Save All Data*, els quals funcionen de manera similar als botons de guardat de gràfiques.

El botó *Save Current Data* guarda a la carpeta on s'ha executat l'aplicació i en format *.txt* els valors numèrics que ha pres l'atribut que s'està visualitzant en aquell moment, d'ençà que ha arrancat l'aplicació, o si no d'ençà que s'ha premut el botó de *Reset*. El *.txt* pren el nom del paràmetre, l'atribut i un *time stamp*.

Així mateix, el botó *Save All Data*, tal com fa el botó *Save All Charts*, guarda en un fitxer *.txt* tots els valors numèrics dels atributs que s'han consultat en algun moment d'ençà que s'ha arrancat l'aplicació. Aquest *.txt* prendrà el nom de *AllData* i un *time stamp*. Amés en el seu interior hi haurà separadors per indicar a quin paràmetre i atribut correspon cadascuna de les dades numèriques.

A la imatge 12 podem veure on estan ubicats els botons.

11. Implementació

Un cop explicat què és el que fa l'aplicació és important saber també com ha estat implementada, i això és el que pretén aquest apartat. Explicar en detall el codi de l'aplicació.

11.1. Implementació amb *Nodetool*

Com ja s'ha esmentat al llarg del treball, inicialment es va planificar fer aquest treball usant *Nodetool* com a eina base per extreure les dades de *Cassandra* i després tractar-les per acabar amb un resultat igual o similar al que s'ha obtingut, però després d'investigar l'eina i d'un temps de desenvolupament es va arribar a la conclusió que no era el més adequat per al que requeria el treball. Així doncs, és convenient explicar, abans d'entrar en detall amb la verdadera implementació, què es va arribar a fer amb *Nodetool* i perquè es va arribar a la conclusió que era necessari canviar a *JConsole* i els *MBeans*.

Nodetool funciona amb comandes de terminal que retornen diferents dades, aquestes comandes donen la informació en "packs" indivisibles i, per tant, en moltes ocasions, per obtenir un paràmetre s'obtenen uns quants més que no es necessiten.

Inicialment, es va fer una investigació de quines eren les comandes que podien ser rellevants per aquest treball i se'n va fer una selecció. Tot seguit es va crear un *script* que fes les crides de forma automàtica i guardes els resultats en un fitxer que posteriorment es pogués tractar.

Un cop es va fer *l'script* i es va executar, es va veure que aquest era molt lent, tardava varis segons en retornar els resultats, i per una aplicació que fa consultes periòdiques i que vol resultats en temps real això era un problema.

Així doncs, en veure que l'eina escollida era lenta i a més no permetia consultar exclusivament allò que es necessitava, sinó que retornava informació addicional innecessària, es va decidir aprofundir un nivell més i fer les consultes directament amb els *MBeans*.

11.2. Classes

En primer lloc, i per tenir una idea global sobre què conté el codi cal saber quines classes hi ha i què fa cadascuna. Així doncs, les classes que formen l'aplicació són les següents:

1. **Principal.java:**

   Aquesta classe és la que conté el *main* i junta totes les altres, es fa la lectura del *YAML*, la connexió a *Minerva*, les crides a funcions d'altres classes i la creació de la interfície gràfica. A més en aquesta està implementada una *interface* que s'utilitza per tenir variables globals que siguin accessibles des de les altres classes.
2. **DataConfig.java:**

   Aquesta classe junt amb la de *Parametro.java* serveixen com a estructures per emmagatzemar i formatejar les dades del YAML
3. **Parametro.java**
4. **InterfazGrafica.java:**

   Classe que conté tot allò relacionat amb la Interfície gràfica i la seva creació i manipulació.

   També conté la creació de les gràfiques, ja que aquestes van incrustades a la interfície.

   Aquesta classe va acompanyada d'un *.form* amb el mateix nom.
5. **AllMemtablesHeapSizeClass.java**
6. **CompletedTasksClass.java**
7. **ExceptionsClass.java**
8. **KeyCacheHitRateClass.java**
9. **KeyCacheHitsClass.java**
10. **KeyCacheMissesClass.java**
11. **KeyCacheRequestsClass.java**
12. **ReadLatencyClass.java**
13. **ReadTimeOutsClass.java**
14. **ReadTotalLatencyClass.java**
15. **ReadUnavailablesClass.java**
16. **RowCacheHitRateClass.java**
17. **RowCacheHitsClass.java**

18. **RowCacheMissesClass.java**
19. **RowCacheRequestsClass.java**
20. **TotalDiskSpaceUsed.java**
21. **WriteLatencyClass.java**
22. **ReadTimeOutsClass.java**
23. **ReadTotalLatencyClass.java**
24. **ReadUnavailablesClass.java**

La resta de classes (de la 5 a la 24) són classes utilitzades per llegir i manipular les dades relacionades amb el paràmetre del mateix nom que la classe.

11.3. Explicació del codi

Tot i que en aquest apartat només s'explicarà el codi final, cal destacar que el procés va ser progressiu i que per a cada fase del desenvolupament (recollida de dades, *YAML*, interfície gràfica) es va tenir una aplicació funcional. Primer mostrant tots els paràmetres per terminal de forma numèrica, després podent escollir quins mostrar amb el *YAML* i finalment amb la interfície gràfica, a la qual després d'aconseguir que funciones adequadament se li van afegir noves funcionalitats com els botons de guardar tot o les etiquetes de temps.

A continuació s'explica el codi en l'ordre en què s'executa quan s'inicia l'aplicació, fent salts entre classes quan sigui necessari i inserint captures de codi quan calguin per clarificar l'explicació.

A més, donat que tots els paràmetres funcionen igual i, per tant, les seves classes són extremadament similars, per clarificar l'explicació, aquesta només es farà per un dels paràmetres. El codi complet es troba a *Github.* (16)

11.3.1. Lectura del *YAML* i obtenció dels valors.

En arrancar l'aplicació el primer que es fa és cridar al *main* que com ja s'ha esmentat es troba a la classe *Principal.java*.

```
public static void main(String[] args) throws IOException, ReflectionException, MalformedObjectNameException,
        InstanceNotFoundException, IntrospectionException, AttributeNotFoundException, MBeanException, InterruptedException {
    //omplir l'array de tabs a eliminar de la GUI de false (iniciailzar)
    Arrays.fill(globales.EliminarTab,  val: false);
    globales.cerrar[0] = false;

    //Lectura del ficher YAML
    InputStream inputStream = new FileInputStream(new File( pathname: "configMonitorizador.yaml"));
    Yaml yaml = new Yaml(new Constructor(DataConfig.class));
    DataConfig data = yaml.load(inputStream);
```

Imatge 13: *Inici del main al fitxer Principal.java*

El primer que fa el *main* és inicialitzar dos paràmetres globals, el primer (*EliminarTab*) serveix per indicar quins paràmetres caldrà mostrar a l'aplicació segons s'indiqui al *YAML*, el segon (*cerrar*) té com a objectiu parar l'aplicació (ja que aquesta, com es veurà més endavant consta d'un *while* infinit.

Tot seguit es fa la lectura del *YAML* fent ús de classes pròpies de *Java* i de la classe *DataConfig.java*. Primer, es crea un *InputStream* on guardar el fitxer *YAML*, a continuació es formata l'entrada per tal que es guardi en un objecte de tipus *DataConfig*. Tot això es pot veure a la imatge 13.

```
public class DataConfig {
    private String nodo;
    private Parametro[] parametros;

    public DataConfig() {

    }
```

Imatge 14: *Atributs i creadora de la classe DataConfig.java*

La classe *DataConfig* consta de dos paràmetres, el nom del node al que connectar-se, i un *array* de *Parametros*, els quals s'expliquen a continuació. A més consta també dels constructors, *getters* i *setters* corresponents. Tal descripció es pot observar a la imatge 14.

```java
public class Parametro {
    private String name;
    private int num;
    private String[] atributs;

    public Parametro () {

    }
```

Imatge 15: *Atributs i creadora de la classe Parametro.java*

La classe *Parametro.java*, per altra banda, consta d'un *String* que indica el nom del paràmetre, un enter que guarda l'índex del paràmetre i un *array* amb els atributs que es volen consultar d'aquest paràmetre. Igual que *DataConfig.java*, també consta de constructores, *getters* i *setters*. Se'n pot veure el codi a la imatge 15.

```java
//creació de la conexió al node de minerva seleccionat en el YAML
JMXConnector JMXMinerva = ConnexioMinerva(data.getNodo() , portNum: "7199");
MBeanServerConnection MC = JMXMinerva.getMBeanServerConnection();


//selecció de quins tabs no eliminar de la GUI
for(int i = 0; i < data.getParametros().length; i++) {
    globales.EliminarTab[data.getParametros()[i].getNum() - 1] = true;
}
```

Imatge 16: *Creació de la connexió a Minerva al main. Fitxer Princial.java*

```java
//conexión a un nodo de Minerva
public static JMXConnector ConnexioMinerva(String hostname, String portNum) throws IOException {
    JMXServiceURL url = new JMXServiceURL("service:jmx:rmi:///jndi/rmi://" + hostname + ":" + portNum + "/jmxrmi");
    return JMXConnectorFactory.connect(url);
}
```

Imatge 17: *Funció ConnexióMinerva. Fitxer Principal.java*

Tot seguit, i tornant al *main,* es troba la creació de la connexió a *Minerva* mitjançant els *MBeans*. La creació es fa mitjançant la funció auxiliar *ConnexioMinerva* que es pot veure a la imatge 17. Aquesta connexió servirà més endavant per accedir a cadascun dels *MBeans* vinculats als paràmetres de *Cassandra*.

Després de la creació de la connexió hi ha un bucle for que emmagatzema a *l'array* de Booleans, *EliminarTab* quins paràmetres es voldran consultar, guardant en la seva posició de *l'array* (d'acord amb l'índex que acompanya el paràmetre en el *YAML*) un *true* en cas que es vulgui consultar. Tot això es pot veure a la imatge 16.

```
//equest bucle s'executa par cada un dels parametres escollits al YAML que s'emmagatzemen a la variable data
//el bucle serveix para obtenir els atributs de cada un dels paràmetres escollits, mitjançant la funció
//getAtributs de cada una de les classes
for (int i = 0; i < data.getParametros().length; i++) {
    int opcion = data.getParametros()[i].getNum();
    switch (opcion) {

        //datos AllMemtablesHeapSize
        case 1:
            AllMemtablesHeapSizeClass AMHS = new AllMemtablesHeapSizeClass(MC);
            globales.attr[opcion-1] = AMHS.getAtributsAllMemtablesHeapSize(data.getParametros()[i]);
            break;

        //datos CompletedTasks
        case 2:
            CompletedTasksClass CT = new CompletedTasksClass(MC);
            globales.attr[opcion-1] = CT.getAtributsCompletedTasks(data.getParametros()[i]);
            break;
```

Imatge 18: *Obtenció dels atributs al main. Fitxer Principal.java*

Després, i com es pot veure a la imatge 18 s'arriba a un bucle *for*, que recorre tots els paràmetres que s'han guardat en llegir el *YAML* i per cadascun d'ells en crea un objecte del seu tipus i es crida a la funció *getAtributs* corresponent (Com ja s'ha comentat a l'inici de l'apartat, l'explicació es farà només per un atribut, l'atribut *AllMemtablesHeapSize*).

Així doncs, ara cal explicar la classe *AllMemtablesHeapSizeClass.java.*

```
public class AllMemtablesHeapSizeClass implements globales {
    private ObjectName urlAllMemtablesHeapSize = null;
    private MBeanInfo MBeanAllMemtablesHeapSize = null;
    private MBeanAttributeInfo[] AllMemtablesHeapSizeAtributes = null;
    private MBeanServerConnection MC = null;

    public AllMemtablesHeapSizeClass(MBeanServerConnection CM) throws MalformedObjectNameException, ReflectionException,
            InstanceNotFoundException, IntrospectionException, IOException {

        urlAllMemtablesHeapSize = new ObjectName("org.apache.cassandra.metrics:type=Table,name=AllMemtablesHeapSize");
        MBeanAllMemtablesHeapSize = CM.getMBeanInfo(urlAllMemtablesHeapSize);
        AllMemtablesHeapSizeAtributes = MBeanAllMemtablesHeapSize.getAttributes();
        MC = CM;
    }
```

Imatge 19: *Atributs i creadora de la classe AllMemtablesHeapSizeClass.java*

Aquesta consta de diversos atributs relacionats amb els *MBeans*, que seran necessaris per accedir-hi i recollir les dades, i la creadora com es pot veure a la imatge 19.

A més també hi ha la funció que s'ha comentat que es crida des del *main, getAtributsAllMemtablesHeapSize*, a la qual se li passa un paràmetre i se n'obtenen els atributs que es volen consultar, els quals s'emmagatzemen a dues variables, la primer, *atributs,* on es guarden els índexs, i la segona, *attrNames,* on es guarden els noms. En ambdós casos, es guarden tots junts en un mateix *String* i separats per comes.

El codi de la funció es pot observar a la imatge 20.

```
//es passa el parametre que es vol consultar per obtenir els atributs a consultar
public String getAtributsAllMemtablesHeapSize(Parametro p) {
    //inicialitza atributs
    String atributs = "";

    //Array amb els noms dels atributs que es volen consultar del parametre
    String[] atrCons = p.getAtributs();

    //en aquest bucle es comparen els noms dels atributs del parametre amb els noms dels atributs que es volen consultar
    //i en cas de que coincideixi es guarda el seu index (la j) al vector atributs
    //a mes a mes es guarden tots els noms dels atributs que es volen consultar en un sol string separat per comes (attrNames)
    //el qual es un array que te una posició per cadascun dels parametres posibles
    for (int i = 0; i < atrCons.length; i++ ) {
        for (int j = 0; j < AllMemtablesHeapSizeAtributes.length; j++) {
            if (atrCons[i].equals(AllMemtablesHeapSizeAtributes[j].getName())) {
                atributs += j;
                atributs += ",";
                globales.attrNames[0] += AllMemtablesHeapSizeAtributes[j].getName();
                globales.attrNames[0] += ",";
            }
        }
    }
    return atributs;
}
```

Imatge 20: *Funció getAtributsAllMemtablesHeapSize. Fitxer AllMemtablesHeapSizeClass.java*

Un cop obtinguts els atributs que es volen consultar de cadascun dels paràmetres, es torna al *main* i es crea la interfície gràfica, la qual s'explicarà al següent apartat per facilitar la comprensió. Després que la interfície sigui creada s'arriba a un bucle *while* infinit, on l'execució es mantindrà fins que es tanqui l'aplicació mitjançant la creueta de la interfície gràfica. (vegeu imatge 21)

```
        //creació de la GUI

        InterfazGrafica frame = new InterfazGrafica( title: "Cassandra Performance Monitor");
        frame.setVisible(true);

        //aquest bucle consulta per cada parametre seleccionat al YAML, el valor de cada atributo seleccionat al YAML
        //i processat al bucle anterior, i li passa els valors a la interfície gràfica perque pugui crear
        // les gràfiques i actualitzar l'etiqueta de valor actual. Està separat de l'anterior para poder posar
        //aquest en un bucle while infinito sense haber d'estar llegint el YAML constantement.
        while (!globales.cerrar[0]) {
            for (int i = 0; i < data.getParametros().length; i++) {
                int opcion = data.getParametros()[i].getNum();
                String atrCons = "";
                switch (opcion) {

                    //datos AllMemtablesHeapSize
                    case 1:
                        AllMemtablesHeapSizeClass AMHS = new AllMemtablesHeapSizeClass(MC);
                        AMHS.consultaAllMemtablesHeapSize(globales.attr[opcion - 1]);
                        atrCons = frame.getAtrConsultat( i: 1);
                        frame.actualitzaLabel(AMHS.getValActualAMHS(atrCons), i: 1);

                        break;
```

Imatge 21: *Creació de la Interfície gràfica i inici del bucle while infinit. Fitxer Principal.java*

Dintre del bucle *while* infinit hi ha un bucle *for*, que s'executa un cop per cada paràmetre a consultar (només els escollits al *YAML*). En aquest es torna a accedir a la classe del paràmetre (en el cas d'aquesta explicació, *AllMemtablesHeapSize*) per obtenir els valors de cadascun dels atributs cridant a la funció *consulta*, la qual es cridarà periòdicament per poder actualitzar la gràfica de forma dinàmica.

```
public void consultaAllMemtablesHeapSize (String AtrConsultar) throws ReflectionException, AttributeNotFoundException,
        InstanceNotFoundException, MBeanException, IOException {

    String[] attrSeparats = AtrConsultar.split( regex: ",");
    String[] nameAttrSeparats = globales.attrNames[0].split( regex: ",");

    for (int j = 0; j < attrSeparats.length; j++) {

        String a = "";
        a += MC.getAttribute(urlAllMemtablesHeapSize, nameAttrSeparats[j]);
        String[] partes = a.split( regex: "E");
        globales.dcd[0][Integer.parseInt(attrSeparats[j])].addValue(Double.parseDouble(partes[0]),nameAttrSeparats[j],String.valueOf(qtyVal[0][j]));
        qtyVal[0][j]++;
        ValAct[0][j] = a;
    }
}
public String getValActualAMHS(String AtrCons) throws ReflectionException, AttributeNotFoundException,
        InstanceNotFoundException, MBeanException, IOException {

    return String.valueOf(MC.getAttribute(urlAllMemtablesHeapSize,AtrCons));
}
```

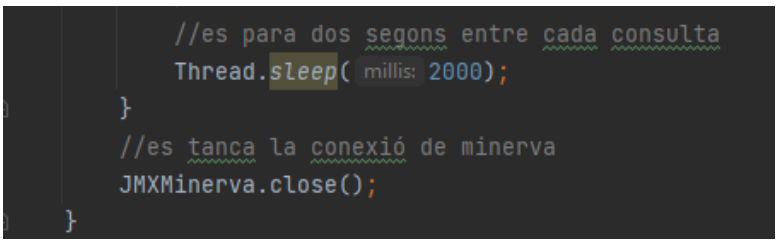
Imatge 22: *Funcions consultaAllMemtablesheapSize i getValActualAMHS. Fitxer AllMemtablesHeapSizeClass.java*

Aquesta funció es pot observar a la imatge 22 on es veu que es recuperen els atributs a consultar del paràmetre concret, mitjançant l'accés als *MBeans* s'obté el valor de cadascun dels atributs desitjats i s'emmagatzema dit valor (juntament amb el nom de l'atribut i quantes dades d'aquest atribut s'han recollit) a la variable global *dcd*.

Aquesta és un *array d'arrays* del tipus *Default Category Dataset* (el necessari per crear les gràfiques) que té un *array* per cada paràmetre, el qual conté una "casella" per cada atribut.

A més s'emmagatzema a *qtyVal* la quantitat de valors que s'han recollit de l'atribut fins al moment, i a *ValAct* el valor més recent que ha pres l'atribut.

Finalment, a la imatge 22 també podem veure la funció *getValAct* que també retorna el valor que pren un paràmetre concret en el moment que es crida. Aquesta s'utilitza per actualitzar l'etiqueta *Current Value* de la interfície gràfica.



Imatge 23: *Final del bucle while infinit, i tancament de la connexió Minerva. Fitxer Principal.java*

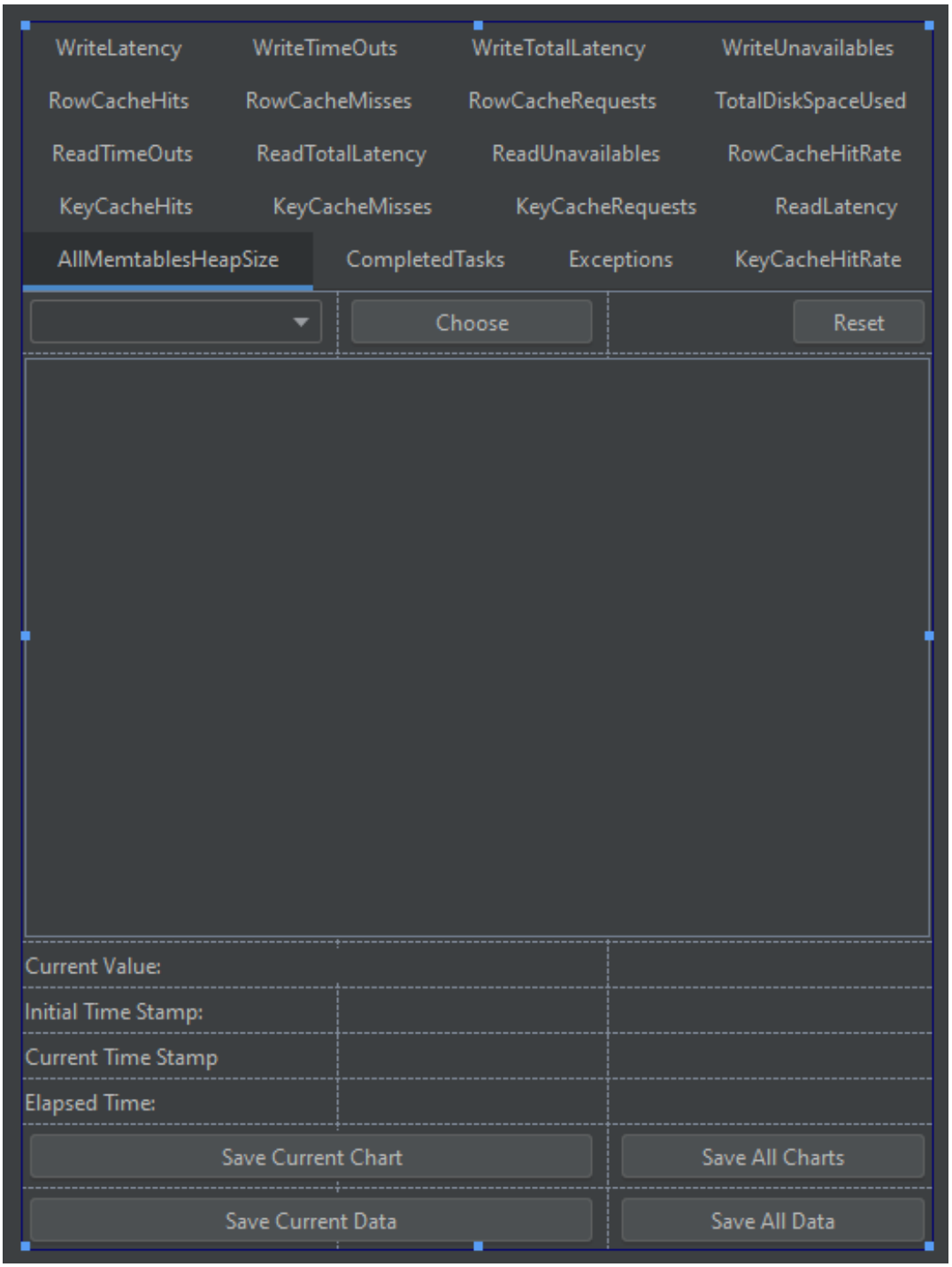
Per acabar, es torna al *main,* després del bucle *for*, es fa un *sleep* de dos segons per espaiar les consultes entre si, i finalment, quan es tanca l'aplicació es fa un tancament de la connexió amb *MBeans* a la base de dades. Això es pot observar a la imatge 23.

11.3.2. Interfície gràfica

Un cop explicat com s'obtenen els valors de cadascun dels atributs, cal explicar la interfície gràfica, la qual està gairebé tota continguda en dos fitxers, el fitxer *InterfazGrafica.form* en el que s'ha fet el disseny de la interfície, i la classe *InterfazGrafica.java* on es descriu el comportament de cadascun dels components.

Així doncs, partint de la crida del *main* que crea la interfície (la qual es pot observar a la imatge 21) se salta als dos fitxers esmentats.

En primer lloc, el *.form*, aquest, com es pot veure a la imatge 24, consta d'un *JTabbedPane* que conté un *JPanel* per cada un dels possibles paràmetres que es poden consultar amb aquesta aplicació. A més, cadascun dels *JPanel* conté els elements esmentats a l'apartat 10.2. d'aquesta memòria.



Imatge 24: *.form de la interfície gràfica*

Així doncs, com els elements estan repetits per cadascun dels atributs, un cop més només s'explicarà el funcionament dels elements corresponents a l'atribut *AllMemtablesHeapSize.*

```
public InterfazGrafica(String title) {
    //para ponerle titulo a la pestaña
    super(title);

    //para que se terminen todos los procesos al cerrar la ventana
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    this.setContentPane(mainPanel);
    this.pack();

    //para hacer que al iniciar aparezca de un tamaño razonable
    this.setMinimumSize(new Dimension( width: 960,  height: 640));

    //para hacer que al iniciar aparezca en el centro de la pantalla
    this.setLocationRelativeTo(null);
```

Imatge 25: *Inici de la constructora de la interfície. Fitxer InterfazGrafica.java*

En primer lloc, quan es fa la crida a la creació de la interfície es fa el que es pot veure a la imatge 25.

Se li assigna un títol, s'indica que amb la creueta es tanca tot, es determina el panell principal, se li assignen unes dimissions i es desvincula la localització perquè en arrancar aparegui al centre de la pantalla.

Tot seguit, i tal com es pot veure a les imatges 26 i 27 es creen els *timers* i *time stamps* per poder-los mostrar a les etiquetes de temps.

```
/*Lo relacionado con las labels del tiempo*/
Long datatime = System.currentTimeMillis();
Timestamp timeS = new Timestamp(datatime);

ActionListener acciones = new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        s[0] += 2;
        if(s[0] == 60) {
            s[0] = 0;
            ++m[0];
        }
        if(m[0]==60) {
            m[0] = 0;
            ++h[0];
        }
        actualizarTimeLabel();
    }
};
Timer t = new Timer( delay: 2000, acciones);
t.start();
```

Imatge 26: *Creació de timers i TimeStamps. Fitxer InterfazGrafica.java*

```
private void actualizarTimeLabel() {
    String tiempo = (h[0]<=9?"0":"")+h[0]+":"+(m[0]<=9?"0":"")+m[0]+":"+(s[0]<=9?"0":"")+s[0];
    Long datatimeE = System.currentTimeMillis();
    Timestamp timeE = new Timestamp(datatimeE);

    for (int i = 0; i < 20; i++) {
        if (globales.EliminarTab[i]) {
            switch (i) {
                case 0:
                    ElapsedTimeAMHS.setText("Elapsed Time: "+tiempo);
                    CurrentTSAMHS.setText("Current Time stamp: " + String.valueOf(timeE));
```

Imatge 27: *Funció actualizarTimeLabel. Fitxer InterfazGrafica.java*

En primer lloc, s'agafa el temps actual i es crea un *time stamp* que ens servirà per donar-li valor a l'etiqueta *Initial Time Stamp*, tot seguit es crea un *listener* que calcula el temps transcorregut d'ençà que s'inicia l'aplicació i que a més, amb l'ajuda de la funció *actualitzaTimeLabel* actualitzarà les *etiqueta Elapsed Time* i *Current Time Stamp*.

Aquest *action listener* se li assigna a un *timer* que el cridarà cada dos segons.

Un cop actualitzades les etiquetes de temps s'arriba a un bucle *for* que afegeix els atributs a monitorejar als desplegables (*comboBox*) de cada paràmetre seleccionat al *YAML*. A més, se li dona valor a l'etiqueta *Initial Time Stamp.*

Aquest bucle es pot observar a la imatge 28.

```
//Bucle que añade los atributos a los comboBox de los parametros seleccionados en el YAML
for (int i = 0; i < 20; i++) {
    if (globales.EliminarTab[i]) {
        String [] attrNamesSeparats = globales.attrNames[i].split( regex: ",");

        switch (i) {
            case 0:
                comboBoxAMHS.removeAllItems();
                InitialTSAMHS.setText("Initial Time Stamp: " + timeS);
                for (String attr : attrNamesSeparats) {
                    comboBoxAMHS.addItem(attr);
                }
```

Imatge 28: *Bucle for s'afegeixen els atributs a les comboBox. Fitxer InterfazGrafica.java*

Després d'afegir els atributs a tots aquells paràmetres que sí que ha escollit l'usuari, s'esborren totes les pestanyes dels atributs que no s'han escollit. El codi d'aquest procés es pot veure a la imatge 29.

```
//Bucle para borrar los tabs que no se han seleccionado en el YAML
int i = 0;
int j = 0;
while (i < 20) {
    if (!globales.EliminarTab[i]) {
        this.tabbedPane1.remove(j);
    }
    else j++;
    i++;
}
```

Imatge 29: *Eliminació de les pestanyes de la interfície no desitjades. Fitxer InterfazGrafica.java*

Finalment, queda per explicar el funcionament de tots els botons dels quals disposa l'aplicació, així doncs, a continuació s'explicarà cadascun dels *listeners* corresponents, els quals només s'executen en cas que l'usuari pitgi el botó vinculat.

- **Botó Choose:**

```
chooseButtonAMHS.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String[] IndexAttr = globales.attr[0].split( regex: ",");
        gNum[0] = comboBoxAMHS.getSelectedIndex();
        creadaData[0][Integer.parseInt(IndexAttr[gNum[0]])] = true;
        creadaChart[0][gNum[0]] = true;
        GraficaAMHS.removeAll();

        if(globales.ValAct[0][gNum[0]].contains("E")) {
            String[] partes = globales.ValAct[0][gNum[0]].split( regex: "E");
            jchart[0][gNum[0]] = ChartFactory.createLineChart( title: "AllMemtablesHeapSize Parameter",
                    categoryAxisLabel: "Amount of Data", valueAxisLabel: "Attribute Value (E" + partes[1] + ")",
                    globales.dcd[0][Integer.parseInt(IndexAttr[gNum[0]])]);
        }
        else {
            jchart[0][gNum[0]] = ChartFactory.createLineChart( title: "AllMemtablesHeapSize Parameter",
                    categoryAxisLabel: "Amount of Data", valueAxisLabel: "Attribute Value",
                    globales.dcd[0][Integer.parseInt(IndexAttr[gNum[0]])]);
        }

        ChartPanel cPanel = new ChartPanel(jchart[0][gNum[0]]);
        cPanel.setMouseWheelEnabled(true);
        cPanel.setPreferredSize(new Dimension( width: 500, height: 400));

        GraficaAMHS.setLayout(new BorderLayout());
        GraficaAMHS.add(cPanel, BorderLayout.NORTH);
        pack();
        repaint();

    }
});
```

Imatge 30: *Listener del botó Choose. Fitxer InterfazGrafica.java*

Aquest *listener* que es pot veure a la imatge 30 és el que s'executa quan es pitja el botó *Choose*. I el que fa és el següent.

En primer lloc, agafa l'index de l'opció seleccionada al *comboBox* i ho guarda a la variable *gNum* (que és un *array* que emmagatzema quina opció hi ha triada a cada una de les *comboBox*). Tot seguit marca a les variables *creadaData* i *creadaChart* (que són uns *arrays* de *booleans*) que aquest atribut s'ha consultat almenys un cop, després neteja el *JPanel* en el que es mostra la gràfica.

A continuació hi ha un *if-else*, que serveix per diferenciar entre valors relativament grans i valors molt petits (amb un exponent E negatiu), ja que els petits si no, no es podrien mostrar adequadament a la gràfica.

Així doncs, d'aquells atributs que prenen un valor molt petit se n'agafa només la part decimal rellevant i el seu exponent es mostra en l'etiqueta de l'eix Y.

Un cop feta aquesta distinció, tant a *l'if* com a *l'else* es crea una gràfica de línia (*LineChart*) que s'assigna a la variable *jchart* (matriu que conté un *JFreeChart* per cada atribut possible de cada paràmetre).

Un cop creada la gràfica, aquesta s'assigna a un *ChartPanel*, al qual se li dona una mida i s'assigna al *JPanel* on es mostrarà la gràfica (*GraficaAMHS*).

Finalment, es pinta la gràfica a la interfície per tal que l'usuari la pugui veure.

- **Botó Reset:**

```
ResetButtonAMHS.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String[] IndexAttr = globales.attr[0].split( regex: ",");
        globales.dcd[0][Integer.parseInt(IndexAttr[gNum[0]])].clear();
        globales.qtyVal[0][gNum[0]] = 0;
    }
});
```

Imatge 31: *Listener del botó Reset. Fitxer InterfazGrafica.java*

El botó de *Reset*, tal com es pot veure a la imatge 31, el que fa és esborrar tots els valors que s'haguessin emmagatzemat al *Default Category Dataset* de la variable *dcd* que correspon a l'atribut que està seleccionat al *comboBox*.

A més a més també reinicia el comptador corresponent de la variable *qtyVal*, ja que aquest enregistra quants valors s'han pres, i en reiniciar s'esborren tots.

- **Botó Save Current Chart:**

```
SaveButtonAMHS.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Long datatimeE = System.currentTimeMillis();
        Timestamp timeE = new Timestamp(datatimeE);
        ChartRenderingInfo info = new ChartRenderingInfo(new StandardEntityCollection());
        File f = new File( pathname: "AllMemtablesHeapSize_"+ comboBoxAMHS.getSelectedItem() + "_Chart_" + timeE + ".png");
        try {
            ChartUtilities.saveChartAsPNG(f, jchart[0][gNum[0]], width: 400, height: 400, info);
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
});
```

Imatge 32*: Listener del botó Save Current Chart. Fitxer InterfazGrafica.java*

El botó *Save Current Chart*, acciona el *listener* de la imatge 32, el qual crea un nou fitxer *.png* amb el nom del paràmetre, l'atribut que s'està visualitzant i un *time stamp* i hi guarda la gràfica amb una mida determinada. Aquesta imatge s'emmagatzema a la carpeta on s'hagi iniciat l'aplicació.

- **Botó Save All Charts:**

```
        saveAllChartsButtonAMHS.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { SaveAllCharts(jchart); }
        });
```

Imatge 33: *Listener del botó Save All Charts. Fitxer InterfazGrafica.java*

```
//funció que guarda un .png per  cada gràfica que s'ha consultat al llarg de l'execució, aquest .png reb el nom del
//paràmetre + identificador de gràfica + un time stamp del moment de guardat. Els .png es guarden a la carpeta on s'ha iniciat
//el programa
public void SaveAllCharts( JFreeChart[][] jchartP) {
    Long datatimeE = System.currentTimeMillis();
    Timestamp timeE = new Timestamp(datatimeE);
    for (int i = 0; i < globales.dcd.length; i++) {
        int k = 0;
        for (int j = 0; j < globales.dcd[i].length; j++) {
                if (creadaChart[i][j]) {
                    ChartRenderingInfo info = new ChartRenderingInfo(new StandardEntityCollection());
                    File f;
                    switch (i) {
                        case 0:
                            f = new File( pathname: "AllMemtableHeapSize_Chart"+k+"_"+ timeE +".png");
                            break;
```

Imatge 34: *Funció SaveAllCharts part 1. Fitxer InterfazGrafica.java*

```
                    try {
                        ChartUtilities.saveChartAsPNG(f, jchartP[i][j],  width: 400,  height: 400, info);
                    } catch (IOException ex) {
                        ex.printStackTrace();
                    }
                    k++;
                }
            }
        }

    }
```

Imatge 35: *Funció SaveAllCharts part 2. Fitxer InterfazGrafica.java*

A les imatges 33, 34 i 35 es pot veure el *listener* del botó *Save All Charts* que guarda totes les gràfiques que s'hagin visualitzat almenys un cop d'ençà que s'ha engegat l'aplicació. I les guarda a la carpeta on s'hagi iniciat l'aplicació.

Per a fer-ho accedeix a tots els *JFreeCharts* que s'emmagatzemen a la variable *jchart* i consulta, mitjançant la variable *creadaChart* si la gràfica ha estat visualitzada almenys un cop. En tal cas crea un fitxer *.png* amb el nom del paràmetre que representa, un identificador de gràfica i un *time stamp* i hi inclou la gràfica.

- **Botó Save Current Data:**

```
saveCurrentDataButtonAMHS.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Long datatimeE = System.currentTimeMillis();
        Timestamp timeE = new Timestamp(datatimeE);
        FileOutputStream os = null;
        try {
            os = new FileOutputStream( name: "AllMemtablesHeapSize_" + comboBoxAMHS.getSelectedItem() + "_Data_" + timeE + ".txt");
        } catch (FileNotFoundException ex) {
            ex.printStackTrace();
        }
        PrintStream ps = new PrintStream(os);
        String[] IndexAttr = globales.attr[0].split( regex: ",");
        ps.println("Atribute " + comboBoxAMHS.getSelectedItem() + " values");
        for(int i = 0; i < globales.dcd[0][Integer.parseInt(IndexAttr[gNum[0]])].getColumnCount(); i++) {
            ps.println(globales.dcd[0][Integer.parseInt(IndexAttr[gNum[0]])].getValue( row: 0,i));
        }
    }
});
```

Imatge 36: *Listener del botó Save Current Data. Fitxer InterfazGrafica.java*

De la mateixa manera que el *listener* del botó *Save Current Chart* desa la gràfica que s'estigui visualitzant, el *listener* de la imatge 36, el del botó *Save Current Data*, desa els valors numèrics que ha pres l'atribut fins al moment, o des de l'últim cop que s'ha pitjat el botó de *Reset*.

Així doncs, primer crea un fitxer *.txt* amb el nom del paràmetre, l'atribut i un *time stamp* i tot seguit, accedeix al *Default Category Dataset* de l'atribut que està seleccionat al *comboBox* i en pren els valors, els quals imprimeix d'un en un al fitxer *.txt*.

- **Botó Save All Data:**

```
        saveAllDataButtonAMHS.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { saveAllData(); }
        });
    }
```

Imatge 37: *Listener del botó Save All Data. Fitxer InterfazGrafica.java*

```
//funció que crea un arxiu txt i hi guarda tots els valors numèrics dels atributs que s'han consultat almenys un cop
//al llarg de l'execució (només es crida quan s'apreta el botó SaveAllData)
//l'arxiu resultant consta de separadors amb els noms dels paràmetres i atributs que representen els valors numèrics
//el nom de l'arxiu txt consta també d'un timestamp del moment en que s'ha creat
public void saveAllData() {
    Long datatimeE = System.currentTimeMillis();
    Timestamp timeE = new Timestamp(datatimeE);
    FileOutputStream os = null;
    try {
        os = new FileOutputStream( name: "AllData_" + timeE + ".txt");
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    }
    PrintStream ps = new PrintStream(os);
    for (int i = 0;  i < globales.dcd.length; i++) {
        if (Arrays.stream(creadaData[i]).anyMatch(k -> k == true)) {
            switch (i) {
                case 0:
                    ps.println("-------------Parameter AllMemtableHeapSize---------------");
                    break;
```

Imatge 38: *Funció saveAllData part 1. Fitxer InterfazGrafica.java*

```
            for (int j = 0; j < globales.dcd[i].length; j++) {
                if (creadaData[i][j]) {
                    ps.println("********************" + globales.dcd[i][j].getRowKey(0) + "**********************");
                    for (int k = 0; k < globales.dcd[i][j].getColumnCount(); k++) {
                        ps.println(globales.dcd[i][j].getValue( row: 0,k));
                    }
                }
            }
        }

    }
}
```

Imatge 39: *Funció saveAllData part 2. Fitxer InterfazGrafica.java*

Per acabar el botó *Save All Data*, el codi del qual es pot observar a les imatges 37, 38 i 39, desa tots els valors numèrics en un fitxer d'aquells atributs que s'hagin consultat almenys un cop.

En primer lloc, crea el fitxer *.txt* que pren per nom *AllData* i un *time stamp* i tot seguit recorre, mitjançant un *for* “anidat”, tots els *Default Category Dataset* emmagatzemats a la variable *dcd,* comprova si han estat consultats almenys un cop amb la variable *creadaData* i en cas afirmatiu imprimeix tots els valors un per un al fitxer *.txt.*

A més a més, entre cada paràmetre imprimeix un separador amb el nom del paràmetre per tal d’indicar el canvi entre un i l’altre i a més també inclou separadors entre atributs amb el nom de l’atribut per indicar a qui pertanyen cadascuna de les dades numèriques i facilitar així la consulta.

12. Integració de coneixements

Per a dur a terme aquest projecte s'han aplicat coneixements adquirits al llarg de la carrera, però també hi ha hagut molta feina d'autoaprenentatge.

En primer lloc, els coneixements adquirits a l'assignatura de BD han servit per entendre el funcionament de *Cassandra* i poder fer-hi consultes, que tot i ser *NoSQL*, és similar en molts aspectes a les bases de dades utilitzades a l'assignatura.

També s'han utilitzat els coneixements adquirits a PROP per a l'ús del *GitHub* per al control de versions, l'ús de l'aplicació *IntelliJ IDEA* i la seva integració amb el *GitHub* i la part de la interfície gràfica desenvolupada amb aquesta mateixa aplicació.

Addicionalment, els coneixements de SO i XC han estat útils a l'hora d'utilitzar el terminal de Linux per a accedir a *Cassandra* i fer *scripts*.

Finalment, els coneixements adquirits en assignatures d'arquitectura de computadors i similars, com AC, AC2 o MP han servit per poder valorar quins paràmetres eren útils a l'hora de mesurar el rendiment d'una base de dades.

Per altra banda, l'ús dels *MBeans* (17), alguns coneixements de *Java* (18), l'ús de la llibreria *JFreeChart* (19) i l'ús del *YAML* (20) han estat apresos de forma autodidàctica mitjançant la cerca per internet de guies i tutorials tot i que cursar la carrera ha ajudat al fet que pogués entendre amb més facilitat aquestes guies i tutorials, ja que al llarg de tot el grau he adquirit diferents tècniques i estratègies que m'han facilitat l'aprenentatge.

13. Lleis i Regulacions

Perquè aquest projecte compleixi la legalitat, cal tenir en compte les llicències de *Cassandra*, *MBeans*, *YAML*, *Java* i de *JFreeChart*.

- *Cassandra* esta emparada sota una llicència Apache v2.0. (21)

- Els *MBeans* formen part de les extensions de *Java* (*JMX* o *Java Management Extensions*), les quals formen part de *Java* que està protegit per l'*Oracle Technology Network License Agreement for Oracle Java SE.*

- *YAML* té una llicència Creative Commons CC-BY 2.0. (22)

- *JFreeChart* utilitza una llicència GNU LGPL. (23)

Per altra banda, com aquesta aplicació és d'ús local, no s'emmagatzema cap mena de dada de l'usuari i per tal no cal una regulació de protecció de dades.

Així doncs, seguint les llicències esmentades i tenint en compte que, en el cas d'aquest projecte, no cal una regulació de protecció de dades, el treball compleix tots els requisits legals.

14. Tests

Un cop programada cada funcionalitat de l'aplicació cal validar que aquestes funcionen de la forma desitjada. Aquest apartat té com a objectiu explicar com s'ha validat que tot el que s'inclou a l'aplicació funciona adequadament.

Com s'ha comentat als apartats anteriors, l'aplicació es va desenvolupar a fases i mantenint sempre un producte usable tot i no ser el producte final. Així doncs, les proves que s'han realitzat van acord amb aquest fet, ja que es van realitzar al final de cada una de les fases.

14.1. Recollida de dades

En primer lloc, recordar que el primer que es va fer va ser accedir a tots els atributs sempre i imprimir per terminal els seus valors numèrics.

Per validar que realment s'obtenien els valors que prenien tals atributs i paràmetres es van comparar un a un amb els valors que mostrava l'aplicació *JConsole*, la qual s'ha utilitzat de referència al llarg de tot el treball.

14.2. *YAML*

Un cop ja es recollien els valors dels atributs es va procedir a crear el *YAML* i fer que l'aplicació el pogués llegir per tal de poder seleccionar només un subconjunt dels possibles paràmetres a consultar.

Per validar que el *YAML* funcionava bé, es va provar de descomentar d'un en un cadascun dels paràmetres i cadascun dels atributs, i després es van provar diverses combinacions de múltiples paràmetres i atributs junts incloent una combinació de tot descomentat.

14.3. Aplicació i graficació

Per acabar es van implementar la interfície gràfica i les gràfiques, les quals es van testejar de la següent forma.

En primer lloc, es va comprovar que només apareguessin les pestanyes dels atributs seleccionats al *YAML*, la qual cosa es veu a simple vista. Després per cadascun dels paràmetres, es va revisar que a la seva *comboBox* només hi apareguessin aquells atributs seleccionats al *YAML*.

Tot seguit es va comprovar que el botó de *Choose* crees una gràfica per tots els paràmetres i tots els atributs (les comprovacions es van fer un per un), i que la gràfica que es creés coincidís amb l'atribut seleccionat a la *comboBox*. A més es va mantenir fins al final del projecte que s'imprimissin els valors dels atributs per terminal, ja que així es podia comprovar si la gràfica prenia el valor correcte i si l'etiqueta *Current Value* també.

Igual que amb el botó *Choose*, quan es van afegir a posteriori la resta de botons (*Save Current Chart, Save All Charts, Save Current Data, Save All Data i Reset*) tots ells es van validar d'un en un per a tots els paràmetres i tots els atributs. Per validar-los es van pitjar i es va observar si feien el que s'esperava.

En últim lloc, es va observar si les etiquetes de temps coincidien amb el temps real mitjançant un rellotge i un cronòmetre.

15. Futur del Projecte

Com tot a la vida, l'aplicació que s'ha desenvolupat al llarg d'aquest projecte no és perfecta i se li podrien incloure altres funcionalitats i millores si es disposés de més temps. Algunes de les possibles noves funcionalitats podrien ser:

- Poder escollir a quina carpeta es vol guardar cadascun dels fitxers que es creen amb els botons de l'aplicació (gràfiques i dades) en comptes que es guardin a la carpeta on s'arranca l'aplicació.

- Poder visualitzar més d'un atribut a la vegada podria ser d'ajuda per poder comparar-los en temps real.

- Poder canviar de tipus de gràfica, és a dir, que no totes les gràfiques fossin *LineCharts.*

- Poder monitorejar més d'un node a la vegada (sense haver d'arrancar més d'un cop l'aplicació).

- Afegir nous paràmetres a monitorejar, ja que actualment només hi ha una selecció de 20 paràmetres.

- Afegir que l'aplicació prengui ella sola decisions de configuració d'acord amb les mètriques obtingudes i ho recomani a l'usuari per tal que a aquest li resulti encara més senzill millorar el rendiment del seu *clúster*.

16. Conclusions

16.1. Compliment dels Objectius Inicials

Un cop acabat el projecte es pot afirmar que es compleixen els objectius establerts a l'inici d'aquest. Ja que s'ha desenvolupat una aplicació que permet a l'usuari:

- **Recopilar dades d'execució de forma senzilla mitjançant una interfície gràfica.** S'ha implementat una interfície gràfica que conté només allò que l'usuari necessita, la qual cosa fa que aquesta sigui clara i fàcil d'usar.

- **Consultar aquestes dades un cop acabada l'execució.** La interfície consta de botons que permeten guardar les gràfiques creades durant l'execució i els valors numèrics extrets de *Cassandra*.

- **Visualitzar les dades obtingudes mitjançant gràfiques i poder extreure conclusions.** L'aplicació crea gràfiques amb les dades extretes, les quals permeten a l'usuari visualitzar de forma clara l'evolució del valor de cada un dels paràmetres.

16.2. Valoració Personal

A escala personal, fer aquest treball ha suposat per mi un gran aprenentatge, donat que he hagut d'investigar moltes tecnologies i eines que desconeixia per complet o que tenia molt oxidades i que actualment podria dir que domino amb bastant soltesa i a més agraeixo dominar, ja que em semblen útils de cara al futur.

A més el fet de realitzar un treball d'aquesta envergadura des de zero i tota sola considero que m'ha ajudat a aprendre a organitzar-me, i a organitzar un projecte, pel que fa a pressupost, organització de tasques i programació del codi.

Per altra banda, és cert que si pogués repetir l'experiència des de zero, faria algunes coses diferents. Sobretot en la planificació del temps per cadascuna de les tasques.

Finalment, crec que realitzar aquest treball m'ha servit per validar-me en l'àmbit personal i comprovar que realment he assolit els continguts apresos durant la carrera.

17. Bibliografia

1. **Apache Software Foundation.** Cassandra Documentation-Overview. [En línia] Apache Software Foundation. [Data: 23 / Setembre / 2021.] https://cassandra.apache.org/doc/latest/cassandra/architecture/overview.html.

2. **Oracle.** What Is a Database? [En línia] Oracle. [Data: 22 / Setembre / 2021.] https://www.oracle.com/database/what-is-database/.

3. —. Distributed Databases. [En línia] Oracle. [Data: 23 / Setembre / 2021.] https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch21.htm.

4. **Graph Everywhere.** Bases de Datos NoSQL | Qué son, marcas, tipos y ventajas. [En línia] Graph Everywhere. [Data: 22 / Setembre / 2021.] https://www.grapheverywhere.com/bases-de-datos-nosql-marcas-tipos-ventajas/.

5. **IBM.** What is batch processing? [En línia] IBM. [Data: 25 / Setembre / 2021.] https://www.ibm.com/docs/en/zos-basic-skills?topic=jobs-what-is-batch-processing.

6. **Apache Software Fundation.** Cassandra Documentation - Nodetool. [En línia] Apache Software Fundation. [Data: 22 / Setembre / 2021.] https://cassandra.apache.org/doc/latest/cassandra/tools/nodetool/nodetool.html.

7. **Matson, John.** How to monitor Cassandra performance metrics. [En línia] Datadog, 3 / Desembre / 2015. [Data: 25 / Setembre / 2021.] https://www.datadoghq.com/blog/how-to-monitor-cassandra-performance-metrics/.

8. **Glassdoor, Inc.** Glassdoor. [En línia] - / Octubre / 2021. https://www.glassdoor.es/Sueldos/index.htm.

9. **Sw hosting and C.T. SL.** Cassandra. Instálalo en tu servidor Cloud. [En línia] - / Octubre / 2021. https://www.swhosting.com/es/cloud/app/cassandra.

10. **Tarifasgasluz by Selectra.** Consulta el precio de la luz (€/kWh): tarifas y comparativa. [En línia] 11 / Octubre / 2021. https://tarifasgasluz.com/comparador/precio-kwh.

11. **DataStax.** Apache Cassandra 3.0 for DSE 5.0. [En línia] DataStax, Inc., 13 / Desembre / 2019. https://docs.datastax.com/en/archived/cassandra/3.0/index.html.

12. **Netinbag.com.** ¿Qué es un MBean? [En línia] Netinbag.com. https://www.netinbag.com/es/internet/what-is-an-mbean.html.

13. **Oracle.** Lesson: Introducing MBeans. [En línia] Oracle. https://docs.oracle.com/javase/tutorial/jmx/mbeans/index.html#:~:text=An%20MBean%20is%20a%20managed,that%20needs%20to%20be%20managed..

14. **yaml.org.** [En línia] yaml.org. https://yaml.org/.

15. **Gibb, Robert.** What is YAML? [En línia] Stackpath, 17 / juliol / 2019. https://blog.stackpath.com/yaml/.

16. **Ferrón, Anna Riera.** Repositori TFGFinal. [En línia] 27 / Abril / 2022. https://github.com/annarf24/TFGFinal.

17. **Oracle.** The Java Tutorials, Creating a Custom JMX Client. [En línia] Oracle. https://docs.oracle.com/javase/tutorial/jmx/remote/custom.html.

18. **Eckel, Bruce.** Thinking in Java. *Thinking in Java.* Stoughton, Massachusetts. : Prentice Hall PTR, 2006.

19. **JFree.org.** JFreeChart 1.5.3 API. [En línia] JFree.org. https://www.jfree.org/jfreechart/javadoc/index.html.

20. **Choudhary, Taimoor.** Reading and Writing YAML Files in Java with SnakeYAML. [En línia] StackAbuse, 19 / Febrer / 2021. https://stackabuse.com/reading-and-writing-yaml-files-in-java-with-snakeyaml/.

21. **Apache Software Fundation.** Apache License, Version 2.0. [En línia] Apache, Gener / 2004. https://www.apache.org/licenses/LICENSE-2.0.

22. **Oracle.** Oracle Technology Network License Agreement for Oracle Java SE. [En línia] Oracle, 10 / Abril / 2019. https://www.oracle.com/downloads/licenses/javase-license1.html.

23. **Free Software Foundation.** GNU Lesser General PUblic License. [En línia] Free Software Foundation, Juny / 2007. https://www.gnu.org/licenses/lgpl-3.0.html.

24. **Apache Software Fundation.** What is Apache Cassandra. [En línia] Apache Software Fundation. https://cassandra.apache.org/_/cassandra-basics.html.